IoT PARA MONITORIZACIÓN DE TRÁNSITO PEATONAL MEDIANTE TÉCNICAS DE APRENDIZAJE PROFUNDO
IoT FOR MONITORING OF PEDESTRIAN TRAFFIC THROUGH DEEP LEARNING TECHNIQUES

JAVIER ROMERO PÉREZ

MÁSTER EN INTERNET DE LAS COSAS. FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin Máster en Internet de las Cosas
Convocatoria Septiembre 2019
Calificación: 8

Curso académico 2018-2019

Director:

Gonzalo Pajares Martinsanz

Autorización de difusión

Javier Romero Pérez

3 de Septiembre de 2019

El abajo firmante, matriculado en el Máster en Internet de las Cosas de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: "IoT para monitorización de tránsito peatonal mediante técnicas de aprendizaje profundo", realizado durante el curso académico 2018-2019 bajo la dirección de Gonzalo Pajares Martinsanz en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Resumen en castellano

En este trabajo se aborda el diseño y desarrollo de un sistema centrado en el cómputo del tránsito peatonal a partir de secuencias de vídeo grabadas en un determinado escenario. Se propone una solución conceptual basada en el aprovechamiento de tecnologías IoT. El cómputo de dicho tránsito puede dividirse en dos partes: primeramente se toman las imágenes de la secuencia de vídeo grabada, y en cada una de ellas se aplican una serie de técnicas de procesamiento de vídeo y detección de movimiento. La aplicación de estas técnicas da como resultado una serie de recortes para cada una de las imágenes, estos recortes son zonas donde se ha identificado que hay algún peatón presente. La segunda parte consiste en el uso de una Red Neuronal Convolucional para llevar a cabo la clasificación de los recortes obtenidos previamente. Los recortes son clasificados por la red neuronal en cuatro categorías diferentes (unCaminante, dosCaminantes, corredor y ciclista). La clasificación permite obtener los datos necesarios para llevar a cabo el cómputo del tránsito de peatones en el escenario en el que se grabó la secuencia.

Los datos obtenidos son subidos a una plataforma en la nube, lo que permite que sean descargados por parte de la administración o de cualquier persona que esté interesada en dichos datos, en ambos casos suponiendo que se tiene acceso, principalmente mediante internet.

Por otro lado, en este trabajo también se ha realizado el estudio y predicción a partir de series temporales. Para ello se han aplicado dos modelos diferentes: un modelo autorregresivo y el uso de redes *Long Short-Term Memory* (LSTM).

Palabras clave

Redes Neuronales Convolucionales - Flujo Óptico - Etiquetado de componentes conexas - *Long Short Term Memory* (LSTM) - Tránsito peatonal

Abstract

This work addresses the design and development of a system focused on pedestrian traffic accounting from video sequences recorded in a given scenario. The paper proposes a conceptual solution based on using IoT technologies. Traffic accounting can be divided into two parts: first, the images of the recorded video sequence are captured, and in each of them a series of video processing and motion detection techniques are applied. The application of these techniques results in a series of clippings for each image; these clippings are areas where it has been identified that a pedestrian is present. The second part consists of the use of a Convolutional Neural Network to carry out the classification of the previously obtained clippings. The clippings are classified by the neural network in four different categories (oneWalker, twoWalkers, runner, and cyclist). The classification allows obtaining the necessary data to carry out the pedestrian accounting in the scenario in which the sequence was recorded. The classification allows obtaining the necessary data to carry out the pedestrian accounting in the scenario in which the sequence was recorded.

The obtained data is uploaded to a cloud platform, which allows it to be downloaded by the administration or any person who is interested in such data, assuming in both cases that they have access, mainly through internet.

On the other hand, in this paper, the study and prediction of data from time series have also been carried out. For this, two different models have been applied: an autoregressive model and the use of Long Short-Term Memory (LSTM) networks.

Keywords

Convolutional Neural Networks - Optical flow - Related components labeling - Long Short Term Memory (LSTM) - Pedestrian traffic

Índice general

Agradecimientos

A todos los profesores que me han permitido adquirir conocimiento y a mis compañeros y amigos que me han acompañado en este viaje.

Capítulo 1

Introducción

1.1. Planteamiento general

En los últimos años las técnicas del IoT (*Internet of Things*) han experimentado un progreso significativo con aplicación en diferentes ámbitos de la vida diaria, entre ellos la monitorización de espacios públicos para conseguir una mejor planificación y uso de recursos. Este es el objetivo principal que se aborda en el presente trabajo, donde se plantea una solución conceptual sobre la aplicación de metodologías IoT para monitorizar el tránsito de peatones, corredores y bicicletas en los carriles habilitados al efecto en las instalaciones de Madrid Río en la ciudad de Madrid.

Por IoT se entienden todas aquellas soluciones que se basan en procesar información adquirida localmente mediante sensores (cámaras, por ejemplo) en un sistema inteligente. Dichos sistemas normalmente gestionan datos de otros dispositivos de una forma centralizada, pudiendo decidir combinarlos entre sí o con datos procedentes de otras fuentes, y que se encargan de publicar los datos recopilados para su acceso posterior mediante las conexiones establecidas al efecto, bien con carácter público o a nivel privado por parte de las administraciones involucradas.

Si nos remontamos a 1990, encontraremos el primer dispositivo que puede ser considerado perteneciente a IoT, una tostadora creada por John Romkey que podía ser apagada/encendida a través de Internet[1]. Previamente a esto, ya había autores y profesionales que

apuntaban hacia lo que hoy se conoce como IoT, aún así el término IoT no fue utilizado de manera pública hasta 1999 en una conferencia sobre RFID organizada por Kevin Ashton[1][2]. Se espera que hacia 2021 haya alrededor de 33 mil millones de dispositivos IoT, llegando a alcanzar 1 billón de dispositivos para el año 2035[1].

El IoT tiene numerosas aplicaciones en la vida cotidiana de las personas, una de estas se sitúa en el ámbito de las *smart cities* o ciudades inteligentes. Estas soluciones están diseñadas para ser desplegadas en un entorno urbano, permitiendo mejorar los servicios ofrecidos a los ciudadanos, generando una mejora en su calidad de vida.

Por ciudad inteligente se entiende aquella que utiliza las tecnologías de la información y la comunicación (TIC) para ofrecer mejores infraestructuras y servicios a sus habitantes. El término ciudad inteligente fue utilizado por primera vez en los años 90[3].

En España se cuenta con el Plan Nacional de Ciudades Inteligentes, elaborado en 2015[4], cuyo principal objetivo es mejorar la eficacia con la que las entidades ofrecen servicios relacionados con las tecnologías TIC, además de objetivos propios de tecnologías TIC como son la mejora de la calidad de vida de las personas, reducir el gasto de recursos, etc.

1.2. Motivación y aportación

Como se ha indicado previamente, el presente proyecto propone una solución conceptual con vistas a su posible implantación real en el futuro. Tecnológicamente, el problema se aborda considerando una cámara que captura imágenes relativas a los tránsitos por un punto determinado, procediendo a la identificación del tipo de elemento en tránsito (bicicletas, peatones, corredores), en un instante determinado, mediante técnicas de aprendizaje profundo, concretamente utilizando Redes Neuronales Convolucionales (*Convolutional Neural Networks*, CNN), obviamente bajo la cobertura de la vigente Ley de Protección de Datos en cuanto a preservar la identidad de las personas que aparecen en las imágenes. Una vez obtenidos los datos de tránsito peatonal, es posible la publicación de estos datos con fines de información tanto a nivel de usuario como para uso de las administraciones públicas,

incorporando técnicas de predicción respecto de su uso en determinados momentos.

Estos datos podrían ayudar a los organismos encargados del mantenimiento de dichos sitios públicos a escoger los momentos idóneos para llevar a cabo labores de mantenimiento (cuando haya menor número de gente, por ejemplo), o ayudar a identificar problemas que generen una disminución del tráfico peatonal.

La motivación de este proyecto viene dada por la rapidez y comodidad que ofrecen las tecnologías IoT a la hora de estudiar el tráfico peatonal. Contar con una herramienta que ofrezca agilidad a la hora de llevar a cabo estudios y gestión del tráfico peatonal, resultaría beneficioso, sobre todo teniendo en cuenta la tendencia creciente de la población mundial[5].

En este contexto, la aportación realizada por el presente trabajo se centra por una parte en el estudio de las distintas metodologías a nivel individual, para llevar a cabo su integración en un sistema global bajo el paradigma IoT.

1.3. Soluciones similares

A la hora de controlar el tráfico de peatones, existen una serie de soluciones ya existentes que lo abordan desde diferentes perspectivas. Se pueden encontrar una serie de aplicaciones, como el *People Counter* ofrecido por Mirame.net[6].

También se puede destacar el caso de la ciudad de Dordrecht, aunque en este caso el conteo de personas es una solución que forma parte de otra más amplia, la ciudad asume la forma de una Smart City contando con un ecosistema de diferentes servicios.

A continuación se proporcionan detalles relativos a ambos ejemplos.

People Counter

La compañía Mirame.net[6], fundada en 2001, se encarga de ofrecer una gama de soluciones tecnológicas a clientes europeos. Entre las soluciones que ofrecen se encuentra el *People Counter*.

People Counter es una aplicación que ofrece a propietarios de comercios diferente infor-

mación sobre su clientela. Permite a los propietarios mantener un control de la gente que entra y sale de su tienda, además de diferentes análisis sobre edad, sexo o tendencias de dichos clientes.

La aplicación de conteo está integrada directamente en cámaras de red IP Standard, reduciendo gastos ya que no se consume ancho de banda en la captura y análisis de las imágenes.

Dordrecht

Dordrecht es una ciudad holandesa situada al oeste de los Países Bajos, que cuenta con aproximadamente 118 900 habitantes, según datos de 2015. En esta ciudad, se llevó a cabo un proyecto de investigación para controlar el tráfico tanto de vehículos como de peatones. Detectar el volumen tanto de coches como de peatones permite conocer cuáles son las calles más transitadas y los tiempos de paso de peatones. Este proyecto fue llevado a cabo por parte de Dimitrios Kyrytsis como parte de sus tesis doctoral[7].

Para desarrollar el proyecto, instaló en los cruces de varias calles una serie de equipos *Meshlium IoT Gateways* desarrollados por Libelium que recibían datos de sensores y permitían que estos llegasen a una plataforma para su gestión.

Los dispositivos detectaban las direcciones Mac de smartphones y equipos de manos libres presentes en coches mediante un escaneo de la señal WiFi. A partir de estos datos e información de distancias entre sensores, se calculó la velocidad de movimiento de cada uno de los dispositivos. Dependiendo de la velocidad, los dispositivos se identificaron con como elementos siendo portados en tres tipos distintos de usuarios: peatón, ciclista y vehículo. Esta clasificación permitió observar cuáles eran las calles más transitadas por cada uno de los tipos de usuarios.

El experimento se llevó a cabo entre septiembre y octubre de 2016 con el objetivo de conseguir mejorar la planificación de la ciudad.

Hasta donde se ha podido investigar estas dos son las soluciones más próximas encontra-

das en relación al planteamiento que se realiza en el presente trabajo. Como puede fácilmente deducirse el objetivo es similar, si bien las soluciones aportadas son diferentes, incluyendo el ámbito y contexto de aplicación. Es por ello, que la solución IoT que se plantea propone una solución conceptual ciertamente novedosa.

1.4. Objetivos

El objetivo principal del proyecto consiste en el desarrollo de una solución conceptual IoT para la monitorización del tráfico peatonal en espacios públicos. Este objetivo también incluye el diseño, implementación y pruebas con la aplicación desarrollada. Teniendo en cuenta lo anterior, bajo este objetivo se propone el estudio de los principios básicos a aplicar, la formulación de un concepto tecnológico, así como una prueba de concepto experimental con validación a nivel de laboratorio de la integración modular propuesta, siendo éste, por tanto, el grado de madurez del proyecto que se propone.

El objetivo principal puede desglosarse en una serie de objetivos específicos:

- Diseño de la arquitectura de la aplicación.
- Aplicación de técnicas de análisis del movimiento en secuencias de vídeo. La aplicación monitoriza el flujo de personas caminando, corriendo o en bicicleta en un determinado escenario, es necesario que la aplicación pueda analizar el movimiento de las diferentes personas realizando las actividades indicadas, tras la captura y análisis de la escena.
- Aplicación de técnicas de redes neuronales convolucionales. La aplicación clasifica entre diferentes tipos de peatones, por lo que es necesario contar con un clasificador potente. Para desarrollar este clasificador se emplean redes neuronales convolucionales. En este objetivo se incluye el aprendizaje y la implementación de este tipo de redes.
- Desarrollo de un sistema de análisis de resultados en la nube. La aplicación proporciona una serie de resultados sobre la monitorización que lleva a cabo; estos resultados son subidos a una plataforma en la nube que permite realizar un análisis exhaustivo sobre

ellos. El análisis que se plantea en este objetivo comprende el uso de técnicas de predicción, bien en la nube o a nivel local.

- Integrar los diferentes módulos y tecnologías para conformar la aplicación IoT, incluyendo la validación de la solución conceptual propuesta a nivel de laboratorio

1.5. Organización de la memoria

La memoria se ha organizado en capítulos, siendo el contenido del resto de capítulos el siguiente. En el segundo capítulo se lleva a cabo una descripción de los métodos y técnicas usadas para procesar imágenes, además de describir también el método de predicción utilizado. El capítulo tres describe la solución que se ha planteado. En el capítulo cuatro se presentan los resultados que se han obtenido, además de un análisis de los mismos. Por último, en el capítulo cinco se presentan las conclusiones y el trabajo que quedaría de cara al futuro.

Capítulo 2

Revisión y descripción de métodos aplicables

2.1. Introducción

En este capítulo se describen los diferentes métodos y técnicas que se han aplicado para la realización de este trabajo desembocando en la aplicación conceptual integrada que se propone. En primer lugar se estudia el procesamiento de imágenes pertenecientes a una secuencia para la identificación, mediante técnicas de detección de movimiento, de peatones y ciclistas, además de delimitar la región que ocupan. Una vez que se han identificado estas estructuras, se procede a realizar una clasificación dependiendo de si son peatones o ciclistas mediante el uso de Redes Neuronales Convolucionales, en particular, se emplea una red ya pre-entrenada denominada AlexNet[8][9][10][11]. En el proceso de clasificación de la red se generarán datos sobre tráfico peatonal permitiendo que esta información sea publicada en la nube. Por otro lado, se utilizan también modelos de series temporales, concretamente modelos autorregresivos y del tipo *Long-Short-Term Memories* para poder llevar a cabo predicciones y análisis sobre los datos provistos por la red neuronal.

Conviene señalar que la red AlexNet se ha elegido por su buen comportamiento para el problema propuesto sin perjuicio de poder utilizar otros modelos de redes de este tipo. En cualquier caso, dado el diseño modular de la aplicación, siempre es posible el reemplazo de cualquier modelo y además conviene señalar que el objetivo del trabajo no es analizar el

comportamiento de los distintos modelos de redes sino la posibilidad de su integración en el diseño conceptual y modular que se propone.

2.2. Detección de movimiento

Teóricamente la aplicación recibe secuencias de imágenes captadas por cámaras dispuestas en diferentes espacios en los cuales se va a analizar el tráfico peatonal. Para la realización de este trabajo, las grabaciones utilizadas fueron obtenidas mediante una cámara de un dispositivo móvil en un escenario concreto del parque de Madrid Río.

A lo largo de este apartado se estudian las técnicas usadas para poder procesar las secuencias de imágenes recibidas y que permiten analizar el movimiento producido en dichas secuencias, finalizando con la identificación de los objetos en movimiento.

2.2.1. Obtención del flujo óptico

Para llevar a cabo el análisis de las imágenes recibidas es necesario estimar el movimiento que se produce en la secuencia. Para realizar esta estimación se utiliza el flujo óptico.

El flujo óptico requiere de imágenes consecutivas separadas por pequeños intervalos de tiempo. Obtener el flujo permitirá poder determinar la dirección y velocidad del movimiento de todos los puntos presentes en la imagen. Para la realización de este apartado se ha tomado como referencia el libro de Pajares y Cruz (2007)[12].

A la hora de llevar a cabo una estimación de movimiento, uno de los métodos posibles es el basado en el gradiente. El gradiente representa los cambios en los niveles de intensidad de una imagen. El flujo óptico se encarga de reflejar los cambios que las imágenes sufren por el movimiento en un intervalo determinado *dt*.

Se puede alcanzar una ecuación de restricciones para el flujo al observar un cambio de intensidad en una imagen de área δS a lo largo de un tiempo.

$$\frac{\delta}{\delta t}\int_{\delta s} f ds = -\oint f v.ndc + \int_{\delta s} \phi ds \tag{2.1}$$

Aplicando el teorema de divergencia de Gauss a la anterior ecuación, se llega a la que sigue:

$$\frac{\delta f}{\delta t} = -f div(v) - v.grad(f) + \phi \tag{2.2}$$

Para poder resolver la ecuación, normalmente se le aplican dos condiciones:

a) La divergencia de v es igual a cero ($div(v)=0$).

b) La intensidad de cualquier punto de un objeto es constante si cumple las condiciones de movimiento de cámaras o de los objetos; por lo que la generación de intensidad de un píxel ϕ es nula.

Al aplicar estas condiciones sobre la ecuación 2.2, se llega a la siguiente ecuación:

$$\frac{\delta f}{\delta t} + v.grad(f) = 0 \tag{2.3}$$

La ecuación 2.3 puede representarse como un desarrollo en serie de Taylor de primer orden si se realiza la implementación de sus derivadas como diferencias entre las intensidades de píxeles adyacentes. Se puede llegar a la siguiente ecuación cuando se representa una imagen dinámica en función de la posición y el tiempo, y suponiendo que $f(x,y,t)$ representa la intensidad de un píxel en una posición (x,y), en un instante t determinado.

$$f(x+dx, y+dy, t+dt) = f(x,y,t) + \frac{\delta f}{\delta t}dx + \frac{\delta f}{\delta t}dy + \frac{\delta f}{\delta t}dt + O(\delta^2) = f(x,y,t) + f_x dx + f_y dy + f_t dt + O(\delta^2) \tag{2.4}$$

En la ecuación 2.4, se pueden despreciar los términos de orden superior($O(\delta^2)$) siempre que dx,dy y dt tengan valores pequeños, resultando en la siguiente ecuación:

$$-f_t = f_x \frac{dx}{dt} + f_y \frac{dy}{dt} \tag{2.5}$$

Esta ecuación es semejante a la ecuación 2.3. En ambas se percibe que la diferencia de intensidad f, en una misma posición e instante, surge por la diferencia en el nivel de intensidad espacial y velocidad en esa posición respecto al observador.

En la ecuación 2.5 se asume que la intensidad de la imagen va a permanecer constante en una trayectoria s durante el movimiento, lo que da lugar a que $df/ds = 0$. Esto va a llevar

una serie de conclusiones, aunque por lo general se recurre a la consideración de que a lo largo de la trayectoria de movimiento lo que va a permanecer constante será el gradiente. Si se aplica que el gradiente sea constante a la ecuación 2.5, se llega a la siguiente ecuación.

$$\begin{bmatrix} \delta^2 f/\delta x^2 & \delta^2 f/\delta x \delta y \\ \delta^2 f/\delta x \delta y & \delta^2 f/\delta y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\frac{\delta(\nabla f))}{\delta t} \tag{2.6}$$

Si se considera que existe un entorno Ω de vecindad para cada punto del sumatorio, la ecuación 2.6 se puede transformar en la siguiente[13]:

$$\begin{bmatrix} \sum_\Omega f_x^2 & \sum_\Omega f_x f_y \\ \sum_\Omega f_x f_y & \sum_\Omega f_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum_\Omega f_t f_x \\ \sum_\Omega f_t f_y \end{bmatrix} = Av = b \tag{2.7}$$

Si a esta ecuación se le aplica mínimos cuadrados, se llega a la solución final:

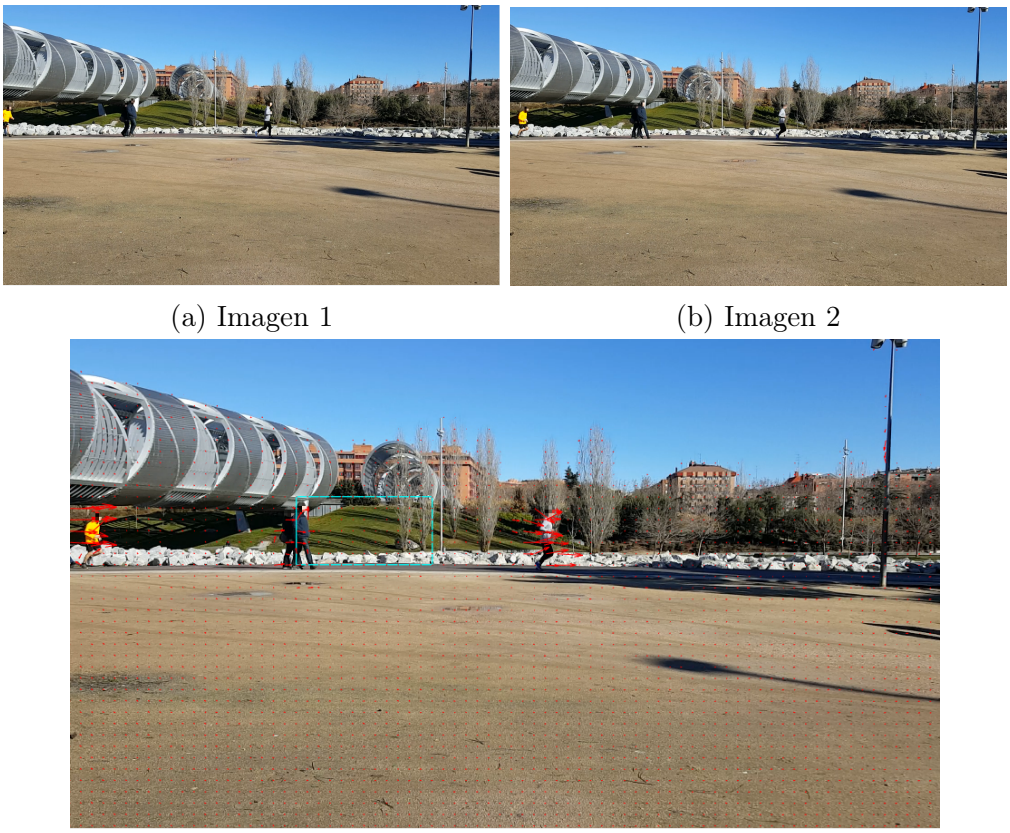
$$v = (A^T A)^{-}1 A^T b \tag{2.8}$$

La figura 2.1 muestra una serie de imágenes ejemplo con dos secuencias consecutivas de una misma serie (a) y (b) para obtener el flujo óptico que se representa en la imagen de la figura (c). Obsérvese la aparición de una serie de flechas sobre los viandantes que representan los vectores de flujo, definidos por sus componentes (u,v) sobre cada píxel de la imagen. Además se identifica el sentido del vector representado por la cabeza de la flecha y su magnitud, que se determina por la longitud de la flecha, de forma que a mayor longitud mayor magnitud y viceversa.

2.2.2. Detección de regiones por movimiento

A partir del flujo óptico obtenido siguiendo las ecuaciones del apartado anterior, se pueden detectar las diferentes regiones presentes en una imagen a partir del movimiento sufrido. Para este apartado, se vuelve a utilizar como referencia el libro de Pajares y Cruz (2007)[12].

Como se ha indicado previamente, el flujo óptico, definido por un vector de componentes (u,v) proporciona un vector para cada uno de los píxeles (x,y) de la imagen. Estos vectores



(a) Imagen 1 (b) Imagen 2

(c) Flujo obtenido

Figura 2.1: Ejemplo de obtención de flujo a partir de dos imágenes consecutivas

cuentan con un módulo y un sentido. A partir del módulo se calcula el valor medio y la desviación estándar, que permiten obtener un umbral determinado. Con dicho umbral se procede a binarizar la imagen, obteniendo como resultado una imagen binaria en blanco y negro.

A partir de esta imagen se lleva a cabo un etiquetado de componentes conexas. Para realizar el etiquetado es necesario partir de una imagen binarizada como la que se ha obtenido previamente. El objetivo del etiquetado es dividir la imagen en regiones con componentes conexas que tengan la misma etiqueta. Por componentes conexas se entienden a todos aquellos píxeles de una imagen que tienen valor binario 1 y que están conectados, mediante un camino, a otros píxeles con valor 1 también. A estas componentes conexas se les asigna una etiqueta que identifica la región de la imagen a la que pertenece cada una.

Para entender mejor el etiquetado, se procede al análisis de la siguiente ecuación donde se lleva a cabo el etiquetado en la matriz a obteniéndose la matriz b.

$$a = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \; b = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 2 \\ 0 & 0 & 1 & 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 1 & 1 & 0 & 2 \\ 0 & 0 & 1 & 1 & 1 & 1 & A \end{bmatrix} \tag{2.9}$$

Para llevar a cabo el etiquetado se utiliza un algoritmo sobre la imagen que va aplicándose fila por fila. El algoritmo asigna etiquetas a los píxeles con valor binario 1 e intenta propagar la etiqueta a los píxeles que se encuentren inmediatamente a la derecha o debajo. En el ejemplo dado en la ecuación 2.9, se le asigna la etiqueta 1 al primer píxel con valor 1 de la primera fila, y al segundo se le asigna la etiqueta 2 ya que no está inmediatamente debajo o a la derecha de ningún píxel con una etiqueta. En la segunda fila, el primer píxel con valor 1 obtiene la etiqueta 1 ya que se encuentra debajo del píxel con dicha etiqueta de la primera fila, mientras que el segundo píxel de valor 1 también obtiene la etiqueta 1 ya que se encuentra a la derecha del píxel ya etiquetado. Este proceso de propagación de etiquetas va repitiéndose en todas las filas de la imagen hasta llegar al último píxel, que tiene la etiqueta A. El valor de este último píxel cambiará según el algoritmo de etiquetado que se emplee.

En el trabajo realizado, se utiliza el procedimiento propuesto por Haralick y Shapiro[14] que implementa lo que se conoce como algoritmo clásico. Este algoritmo realiza dos pasadas a través de la imagen y utiliza una tabla de equivalencias. El primer paso realiza una propagación de etiquetas similar a la que ya se ha explicado. Cuando dos etiquetas diferentes se intentan propagar a un mismo píxel, la etiqueta menor prevalece y la equivalencia se guarda en la tabla de equivalencias. La tabla de equivalencias contiene pares ordenados, donde los valores son las etiquetas equivalentes que se han hallado en la primera pasada. Una vez ha acabado la primera pasada se encuentran las clases de equivalencia a partir de la tabla de equivalencias que se ha creado. A cada una de las clases de equivalencia se les asigna una etiqueta, por lo general la más pequeña. Por último, se realiza una segunda pasada donde a cada píxel se le asigna la etiqueta de la clase de equivalencia.

Una vez ha finalizado el etiquetado, se obtienen una serie de propiedades de cada una

de las regiones. Dentro de esas propiedades destaca lo que se conoce como *bounding box* que determina las áreas candidatas, para obtener estas áreas se comparan dos frames consecutivos y se seleccionan las áreas en las que se ha producido un movimiento significativo y que se encuentren dentro de un área especificada en la imagen, sólo las regiones que se encuentran dentro de esa área son consideradas áreas candidatas. Estas áreas candidatas se utilizan para delimitar zonas en las imágenes y realizar recortes como los de la figura 2.2. Estos recortes son los que se pasarán a la red neuronal para realizar tanto su entrenamiento como la clasificación de dichos recortes.

(a) Ciclista (b) Corredor (c) Caminante

Figura 2.2: Ejemplos de recortes utilizados por la red neuronal para el entrenamiento y clasificación

2.3. Redes neuronales convolucionales

Para llevar a cabo la clasificación de los elementos en movimiento que se encuentran dentro de las secuencias de imágenes se ha optado por utilizar redes neuronales convolucionales (*Convolutional Neural Networks*, CNN).

Las CNN son redes neuronales con una topología de rejilla para el procesamiento de datos, utilizando rejillas 2-D para imágenes. Las CNN se denominan así ya que utilizan la operación de convolución, en vez de la multiplicación de matrices, en al menos una de las capas de la red. Estas capas con operaciones de convolución poseen filtros que a su vez contienen una serie de pesos asociados, estos pesos son los que se van ajustando en la fase de aprendizaje de la red. Las capas de convolución simulan comportamientos de modelos

biológicos de visión, por lo que son de alguna forma similares a lo que se conoce como filtros de Gabor[15 16 17].

Existen diferentes arquitecturas de CNN ya definidas. Por las razones expuestas anteriormente, para la aplicación desarrollada se optó por utilizar AlexNet[11].

En la figura 2.3 se puede observar la estructura de capas de una red AlexNet. AlexNet es un modelo ya entrenado con anterioridad para realizar la clasificación de mil objetos de diferentes categorías. Cuenta con veinticinco capas en las que se aplican diferentes operaciones, incluyendo las de convolución, con núcleos de distintos tamaños. El modelo necesita recibir como entrada imágenes de las diferentes categorías a clasificar, que tengan unas dimensiones de 227x227x3.

En la figura 2.3 se puede observar que cada una de las capas también cuenta con una serie de símbolos que indican las distintas propiedades de cada una de las capas, k indica el número de filtros que se han aplicado, p *(padding)* se refiere a la cantidad de ceros que han sido añadidos y s *(stride)* indica las unidades que se ha desplazado el núcleo. A continuación se describen las diferentes operaciones que se utilizan en la red AlexNet

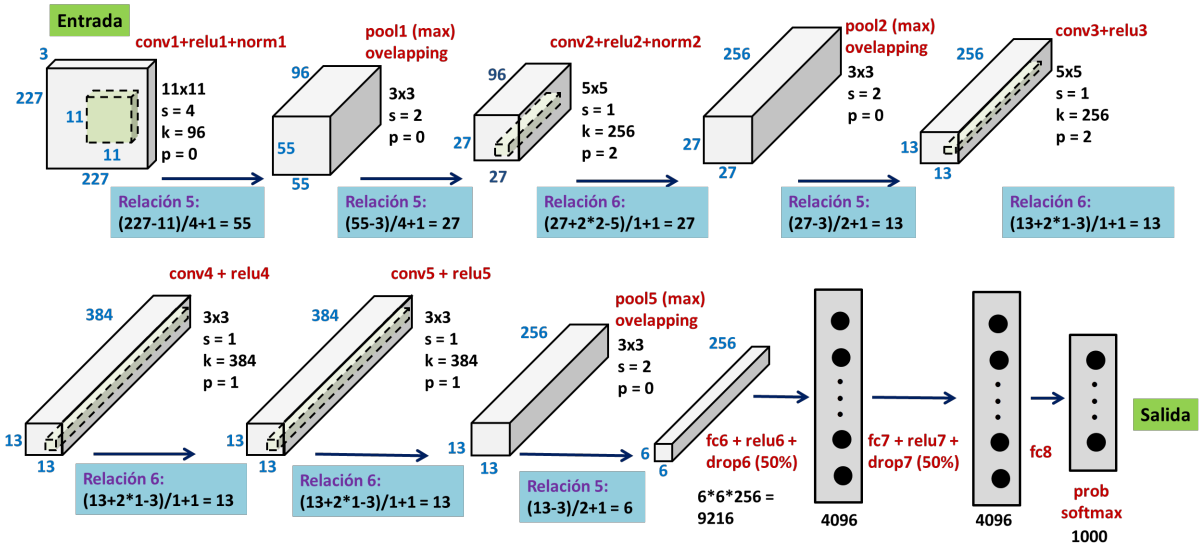


Figura 2.3: Estructura de red Alexnet

2.3.1. Convolución

Por lo general, una operación de convolución involucra dos funciones con valores reales como argumento. Para entender la convolución se puede usar como ejemplo una fuente de luz variable cuya intensidad es recibida por un sensor, que proporciona una salida en una posición x en el tiempo t, $x(t)$. Dado que x y t son valores reales, se pueden obtener diferentes lecturas en diferentes instantes por la variabilidad de la fuente de luz. Además, el sensor puede estar contaminado con cierto ruido, por lo que lo más acertado es realizar un promedio de la salida con varias medidas. También hay que tener en cuenta que las medidas recientes son más relevantes, por lo que se les puede dar más relevancia mediante la función de promediado $w(a)$, donde a representa el alejamiento de la medida en el tiempo. Todo esto lleva a obtener la siguiente función:

$$s(t) = \int x(a)w(t-a)da \tag{2.10}$$

Esta operación se denomina convolución. En el aprendizaje máquina, dado que tanto la entrada como el núcleo (*kernel*) son, por lo general, vectores o matrices multidimensionales de datos y parámetros respectivamente, se utiliza una versión diferente de la ecuación que es la que sigue:

$$S(i,j) = (I * K)(i,j) \tag{2.11}$$

En esta ecuación, el parámetro K es lo que se denomina como kernel de convolución. La salida, representada por S, se corresponde con un mapa de características o *feature map*.

2.3.2. ReLU

La función ReLU (*Rectified Linear Unit*) $f(x) = max(0,x)$, es una función de activación encargada de rectificar linealmente los resultados de la convolución. Atendiendo a su definición, la función ReLU se encargará de devolver 0 para los valores negativos y el valor de entrada para los positivos. La función viene representada en la figura 2.4 y cuenta con las siguientes características: -Gradiente no saturado, por el hecho de que $x > 0$; por lo que el

problema de la dispersión del gradiente en proceso de propagación inversa se ve aliviado y los parámetros en la primera capa de la red neuronal pueden actualizarse de forma rápida. -Baja complejidad computacional; no obstante posee la desventaja de que la neurona ReLU puede morir cuando recibe un gradiente negativo alto durante la retropropagación. Esto puede evitarse inicializando cuidadosamente los pesos.

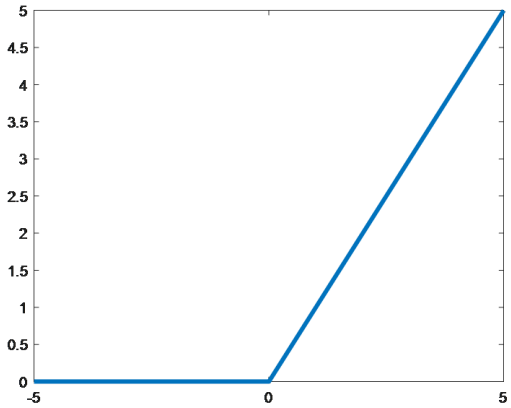


Figura 2.4: Gráfica de función ReLU

2.3.3. Pooling

Las redes neuronales cuentan con capas de *pooling* o agrupaciones. Estas capas proporcionan invarianzas a pequeñas traslaciones que afectan a la entrada. Existen diferentes operaciones de *pooling*, como por ejemplo el máximo (max), que divide la entrada en ventanas, generando a la salida el máximo de la ventana. Otra operación es la media (mean) de la ventana en la que la salida es el resultado de aplicar dicha operación sobre la ventana [18] [19].

Desde el punto de vista de la convolución, la operación de pooling puede interpretarse en base a la hipótesis de que alguna función se va a repetir sobre ventanas de la entrada. A partir de estas relaciones, el *pooling* es una operación sin *zero padding*, de manera que la relación que describe el caso general es:

$$o = [\frac{i-k}{s}] + 1 \tag{2.12}$$

Una capa típica de convolución consta de tres estados. Un primer estado donde se realizan varias convoluciones para generar un conjunto de activaciones lineales. Un segundo estado donde cada activación lineal se ejecuta a través de una función de activación no lineal, como por ejemplo la función de activación rectificada lineal(ReLU). Este estado es conocido a veces como *detector stage*. Y un último estado donde se utiliza la función de *pooling* para modificar la capa de salida, que reemplaza la salida de la red en una cierta localización con una operación estadística involucrando otras salidas cercanas.

En definitiva, el *pooling* ayuda a conseguir que la representación sea invariante ante ligeros desplazamientos de la entrada, es decir si se desplaza la entrada en una pequeña variación, los valores de muchas salidas que hayan sufrido esta operación no sufrirán modificaciones.

2.3.4. Normalización

La operación de ReLU tiene la propiedad de no requerir normalización para prevenir saturaciones[10], sin embargo, se puede aplicar una normalización local como ayuda a la generalización. Si se entiende por $a^i_{x,y}$ la actividad de una neurona al aplicar el núcleo i en la posición *(x,y)* y a continuación se aplica la operación ReLU, la actividad de la respuesta normalizada $b^i_{x,y}$ se representa con la ecuación:

$$b^i_{x,y} = \frac{a^i_{x,y}}{\left(k + \alpha \sum_{j=\text{máx}(0,i-n/2)}^{\text{mín}(N-1,i+n/2)} (a^i_{x,y})^2\right)^\beta} \tag{2.13}$$

En esta expresión la suma se extiende sobre n mapas adyacentes, ordenados arbitrariamente antes de llevar a cabo el aprendizaje, generados por núcleos en la posición (x,y), con un total de N núcleos en toda la capa. El valor de las constantes k, n, α y β presentes en la ecuación 2.13 se pueden determinar mediante el uso de un conjunto de validación, Krizhevsky y col (2012)[10] propone valores de $k = 2$, $n = 5$, $\alpha = 10^{-4}$ y β =0.75.

2.3.5. Capas Dropout

Uno de los problemas que puede surgir en las redes neuronales es lo que se conoce como *overfitting*. En las redes neuronales se ajustan un número elevado de pesos en numerosas neuronas. La figura 2.5 muestra un ejemplo de *underfitting* y *overfitting* frente a un ajuste correcto. La cuestión es qué ajuste es mejor, el que pasa por todos los puntos mediante un polinomio con un grado elevado (derecha) o el que lo hace mediante una línea recta (izquierda). En el primer caso, el modelo se ajusta bien a los datos pero en el momento de la decisión, ésta no es la adecuada.

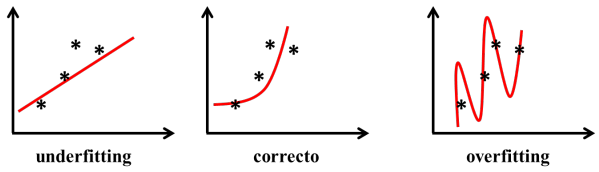


Figura 2.5: Ejemplos de tres ajustes

Un método posible para evitar *overfitting* es aplicar regularización, que consiste en modificar la función objetivo a minimizar añadiendo términos adicionales que penalizan pesos con valores elevados. Así por ejemplo, dada la función objetivo F se le puede añadir el término $\lambda f(\theta)$ de forma que $f(\theta)$ crece a medida que el argumento θ también crece y λ es el coeficiente de regularización. El valor de este parámetro determina la protección contra el overfitting, de forma que si es cero, no se realiza ninguna acción protectora, por el contrario, si es demasiado grande, el modelo dará prioridad a mantener θ lo más pequeño posible, al tratar de encontrar los valores de los parámetros que que mejor se ajustan al conjunto de datos de entrenamiento.

No obstante, en las redes neuronales, para mitigar el efecto del *overfitting* se propone anular (*dropout*) determinado tipo de neuronas incluidas sus salidas para evitar el problema del sobre ajuste durante el proceso de entrenamiento, lo cual previene el *overfitting* de los pesos[20]. La selección de unidades a anular se lleva a cabo de forma aleatoria. La red resultante es la que mantiene las unidades que hayan sobrevivido al *dropout*. Sin embargo,

en lugar de eliminar neuronas, también existe la posibilidad de mantenerlas asignándoles un hiperparámetro de forma aleatoria que es en realidad una probabilidad p de supervivencia durante la fase de entrenamiento, utilizado para disminuir la influencia del peso en la fase de test. En la figura 2.6 se observan nodos cancelados (en rojo) y con el peso de activación atenuado por la probabilidad p durante la fase de test.

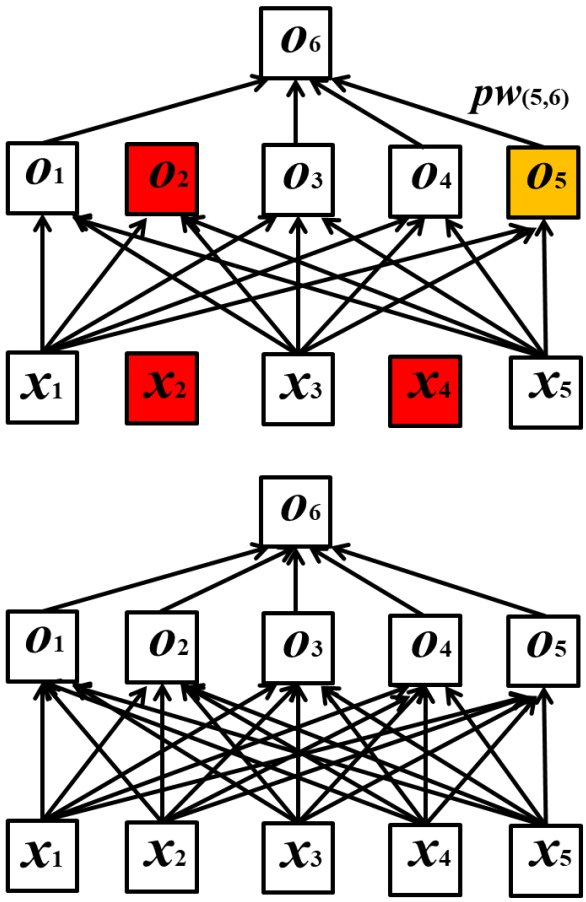


Figura 2.6: Dropout

El *dropout* puede aplicarse tras la salida de una determinada capa, como por ejemplo, tras un *pooling* como se puede observar en la figura 2.7.

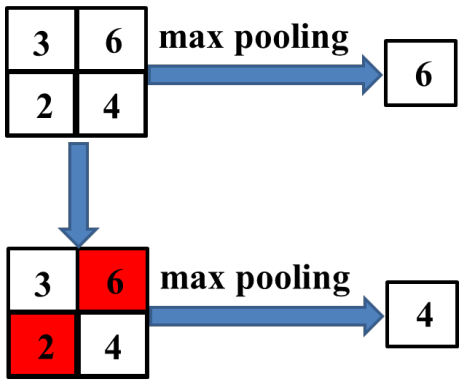


Figura 2.7: Dropout y max pooling

2.3.6. Softmax

La función exponencial normalizada o *softmax* aparece por norma general en las últimas capas ocultas y se define en la ecuación 2.14.

Se emplea para llevar a cabo la proyección de un vector n-dimensional, x de valores reales en un vector n-dimensional *softmax(x)* de valores reales en un rango comprendido entre [0,1].

$$softmax(x)_i = [\frac{exp(x_i)}{\sum_{j-1}^{n} exp(x_j)}] parai = 1, ..., n \tag{2.14}$$

Se aplica la función exponencial a cada elemento x_i del vector de entrada x y se normalizan esos valores por la suma de las exponenciales, lo que asegura que la suma de las componentes del vector de salida $softmax(x) = 1$.

2.3.7. Gradiente descendente

La técnica del gradiente descendente se emplea en la retro-propagación para poder llevar a cabo el ajuste de los pesos de la red. El objetivo es minimizar una función, conocida como función objetivo o loss function, con la siguiente forma:

$$J(w) = \frac{1}{n} \sum_{i=1}^{n} J_i(w) \tag{2.15}$$

El parámetro w es el empleado para minimizar J y debe estimarse. J_i se corresponde con

la i-ésima observación en el conjunto de datos de entrenamiento. $J_i(w)$ es el valor de la *loss function* en el i-ésimo ejemplo y *J(w)* se corresponde con el riesgo empírico. El gradiente descendente es utilizado para minimizar la función de manera iterativa en t iteraciones.

$$w(t+1) = w(t) - \epsilon \frac{1}{n} \sum_{i=1}^{n} \nabla J_i(w) \tag{2.16}$$

De esta manera iterativa el método se encarga de recorrer el conjunto de entrenamiento realizando la actualización anterior para cada una de las muestras. Se puede llevar a cabo más de una pasada sobre el conjunto de entrenamiento hasta que se consiga la convergencia. En caso de realizar más de una pasada, es recomendable seleccionar los datos aleatoriamente, lo que se conoce como criterio estocástico, para evitar ciclos.

2.3.8. Salida de la red

Como se ha mencionado anteriormente, la red AlexNet es una red entrenada previamente para realizar la clasificación de 1000 objetos. En este trabajo se ha limitado la clasificación a solo cuatro categorías que son: unCaminante, dosCaminantes, corredor y ciclista. Tras la ejecución de la capa softmax, la red proporcionará valores para las cuatro categorías, que se corresponderán con la probabilidad de pertenencia de la persona clasificada a cada una de estas categorías. Dependiendo del error que se obtenga respecto a la salida esperada, se llevará a cabo un proceso de retro-propagación para poder ajustar los pesos de la red, cuyo objetivo será minimizar la función de pérdida.

2.4. Series temporales

2.4.1. Definición

Una serie temporal como su nombre indica, se refiere a una serie de valores que son muestreados en el tiempo en intervalos regulares por lo general. Relacionándolo con el trabajo actual y a modo de ejemplo pedagógico, se podría tomar la evolución que se produce en el tráfico de bicicletas en un parque todos los lunes entre las 10 y las 12 de la mañana. Una

serie temporal está formada por N observaciones, que se encuentran ordenadas en el tiempo en base a una determinada variable (univariantes) o varias variables (multivariantes) de una unidad que puede ser observada en distintos instantes[21]. Tienen la siguiente representación:

$$y_t = y_1, y_2, ..., y_N; t = 1, ..., N \tag{2.17}$$

La serie se emplea para la realización de un modelo y una predicción posterior basada en el modelo obtenido y los parámetros que hayan sido estimados mediante dicho modelo. Por modelo de un proceso se entiende un conjunto de hipótesis sobre las propiedades de ese proceso. Para este trabajo, se han empleado dos modelos de series temporales, los cuales se explican a continuación.

2.4.2. Modelo autorregresivo

Los modelos ARMA (*AutoRegressive Moving Average*) son aplicados a series temporales para entender y, en particular, predecir valores futuros de la serie. Entre los diferentes modelos pertenecientes a ARMA, se pueden encontrar los modelos Autorregresivos o AR. Un proceso autorregresivo, AR(p), con orden p cumple la siguiente ecuación:

$$X_t = c + \phi_1 X_{t-1} + \phi_2 X_{t-2} + ... + \phi_p X_{t-p} + \varepsilon_t \tag{2.18}$$

donde c es una constante, $\phi_1, \phi_2..., \phi_n$ se corresponden con parámetros del modelo y ε_t es ruido blanco.

El objetivo en esta clase de procesos es averiguar el orden p del modelo, además de los valores de los parámetros ϕ. Uno de los procedimientos más comunes para lograr esto consiste en la aplicación de las ecuaciones de Yule-Walker[22][23]. El fundamento principal de estas ecuaciones es la existencia de una correspondencia entre los parámetros que se quieren averiguar y la función de covarianza del proceso, esta correspondencia puede ser invertida para averiguar los parámetros a partir de covarianzas. Las predicciones en procesos autorregresivos se logran mediante la aplicación de la regla de la cadena sobre la ecuación 2.18. Cada una de las predicciones que se llevan a cabo sirve a su vez para conseguir las

siguientes predicciones. Estas predicciones se van realizando de forma sucesiva hasta llegar a un límite n previamente definido.

$$\begin{aligned} X_t &= c + \phi_1 X_{t-1} + \phi_2 X_{t-2} + ... + \phi_p X_{t-p} \\ X_{t+1} &= c + \phi_1 X_t + \phi_2 X_{t-1} + ... + \phi_p X_{t-p+1} \\ X_{t+2} &= c + \phi_1 X_{t+1} + \phi_2 X_t + ... + \phi_p X_{t-p+2} \\ &\cdot \\ &\cdot \\ X_{t+n} &= c + \phi_1 X_{t-1+n} + \phi_2 X_{t-2+n} + ... + \phi_p X_{t-p+n} \end{aligned} \tag{2.19}$$

2.4.3. Long Short-Term Memory

El modelo *Long Short-Term Memory* (LSTM) pertenece a un grupo más amplio de redes denominadas redes neuronales recurrentes(RNN, *Recurrent Neural Networks*). Hochreiter y Schmidhuber[24] introdujeron las LSTM con la finalidad de resolver el problema de que los gradientes que se retro-propagan sobre muchos estados tienden a desaparecer, afectando a la optimización.

Las LSTM procesan secuencias de pares de entrada salida $(x, y)_{t-1}^{T}$. Para cada uno de los pares (x_t, y_t), la celda LSTM toma una entrada x_t nueva y el valor oculto h_{t-1} obtenido en el paso anterior y produce una estima $\hat{y}_t$ para la salida y_t a partir de la secuencia previa $x_1, x_2, ..., x_t$, con un nuevo valor oculto h_t y un nuevo valor del estado de la celda o memoria C_t. En la figura 2.8 se muestra la estructura de una celda LSTM sobre el tiempo[25]. En la figura 2.9 se puede observar una estructura característica LSTM[26][27].

El estado de cada celda se corresponde con una línea de transporte que atraviesa la celda y que tiene las interacciones indicadas en la figura 2.10, de manera que la LSTM puede ir eliminando o añadiendo información al estado de la celda mediante estructuras denominadas *gates*, que integran operaciones como la que se muestra en el lado derecho de la figura 2.10.

El primer paso de la red es decidir qué información elimina del estado de la celda, la responsable de esto es una capa sigmoidal que se conoce como *forget gate* (f_t) 2.11, de

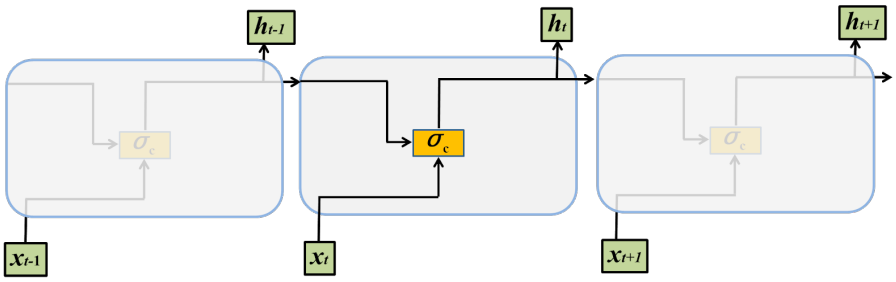


Figura 2.8: Estructura general y conexión de las celdas

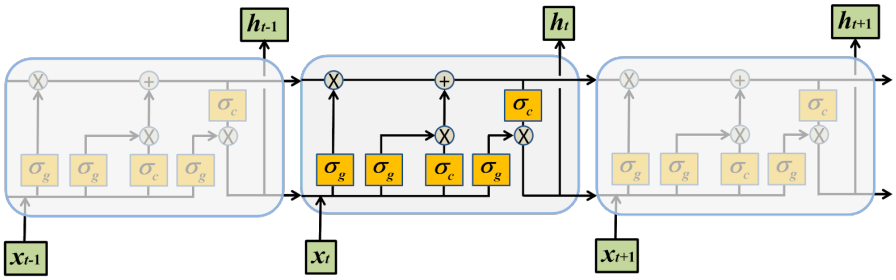


Figura 2.9: LSTM

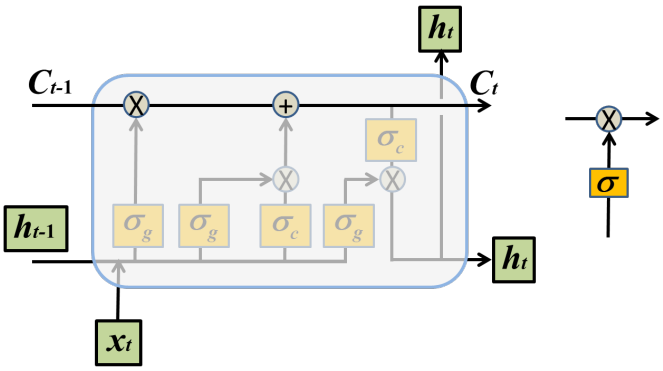


Figura 2.10: LSTM

forma que tomando los pesos (U_{xf}, W_{hf}) y un bias, b_f, a través de la función sigmoide (σ_g) se genera un valor entre 0 y 1 para cada estado de la celda, de manera que un 1 significa mantener ese estado y un 0 deshacerse del estado.

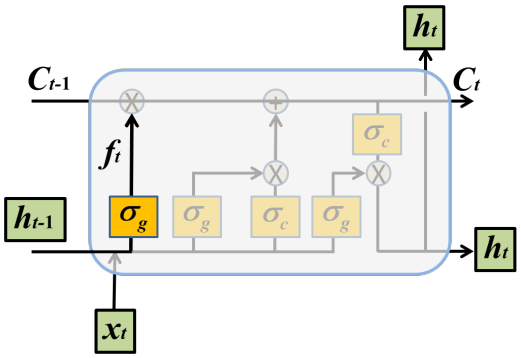


Figura 2.11: Forget gate

$$f_t = \sigma_g(U_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{2.20}$$

El siguiente paso es decidir la información nueva que va a almacenarse en el estado de la celda. Este proceso se divide en dos partes; primero, una capa sigmoide conocida como *input gate* (i_t) 2.12, decide los valores que se van a actualizar. A continuación una capa σ_c, por lo general con función de tipo *tanh*, crea nuevos valores candidatos (g_t) que podrían ser agregados al estado, tomando en cuenta los pesos $(U_{xi}, W_{hi}, U_{xc}$ y $W_{hc})$ y los bias correspondientes (b_i, b_c). El siguiente paso es combinar ambos para llevar a cabo una actualización del estado.

$$\begin{aligned} i_t &= \sigma_g(U_{xi}x_t + W_{hi}h_{t-1} + b_i) \\ g_t &= \sigma_c(U_{xc}x_t + W_{hc}h_{t-1} + b_c) \end{aligned} \tag{2.21}$$

A continuación se debe actualizar el estado de la celda previo, g_{t-1}, al nuevo estado g_t, como se muestra en la figura 2.13. Para ello, se multiplica el estado antiguo por f_t consiguiendo olvidar las cosas que se decidió olvidar previamente. Por último, se suma el término i_t *e* g_t, que son los nuevos valores candidatos.

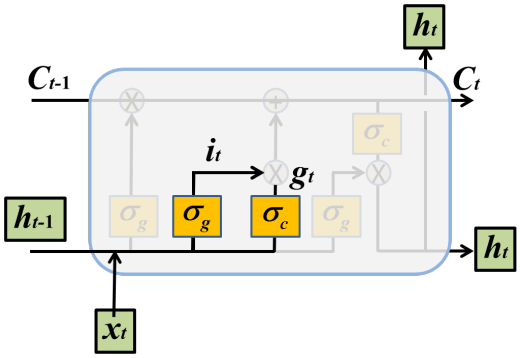


Figura 2.12: Input Gate

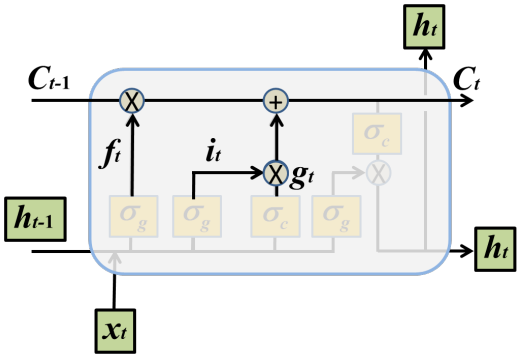


Figura 2.13: Actualización del estado

$$C_t = f_t e C_{t-1} + i_t e g_t \tag{2.22}$$

Finalmente, hay que obtener el resultado de la salida, que se basará en una versión filtrada del estado de la celda. Para ello, primero se aplica la función sigmoidea que decide qué partes del estado de la celda contribuyen a la salida, Luego, se aplica la función *tanh* que se encargará de situar los valores en un rango de -1 y 1, la salida de esta función se multiplica por la salida de la puerta sigmoidea, así solo contribuyen a la salida las partes que se hayan seleccionado.

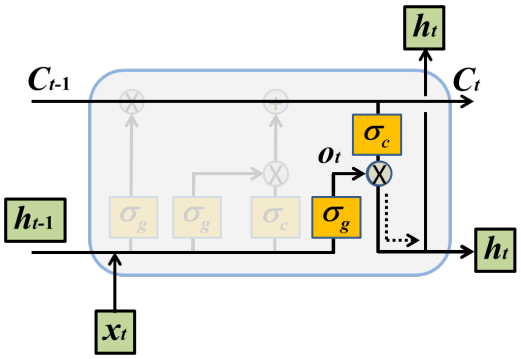


Figura 2.14: Generación de la salida

$$\begin{aligned} o_t &= \sigma_g(U_{xo} x_t + W W_{ho} h_{t-1} + b_o) \\ h_t &= o_t \; e \; \sigma_c(C_t) \end{aligned} \tag{2.23}$$

Capítulo 3

Diseño de la aplicación

3.1. Introducción

En el presente capítulo, en primer lugar, se lleva a cabo una descripción del diseño conceptual de la arquitectura propuesta. A continuación se describe la transferencia e intercambio de datos entre los diferentes módulos que componen la arquitectura. Por último, se explica cómo se desarrolla el modelo de simulación aplicado. Con el fin de verificar la validez de la solución de concepto se plantea y se describe el correspondiente proceso de simulación debido a la imposibilidad, tanto material como temporal, de poder implantar este sistema en ubicaciones físicas.

3.2. Arquitectura

En la figura 3.1 se representa un diagrama conceptual de la arquitectura propuesta en este trabajo. El diagrama cuenta con diferentes módulos cuya descripción se realiza a continuación.

- Escenario: Lugar donde se desarrolla la acción.
- Peatón: Persona física que se encuentra en el espacio público observado. Para la realización de este trabajo, se distinguen cuatro tipos catalogados como peatones: caminantes, doscaminantes (personas caminando en pareja), corredores y ciclistas.

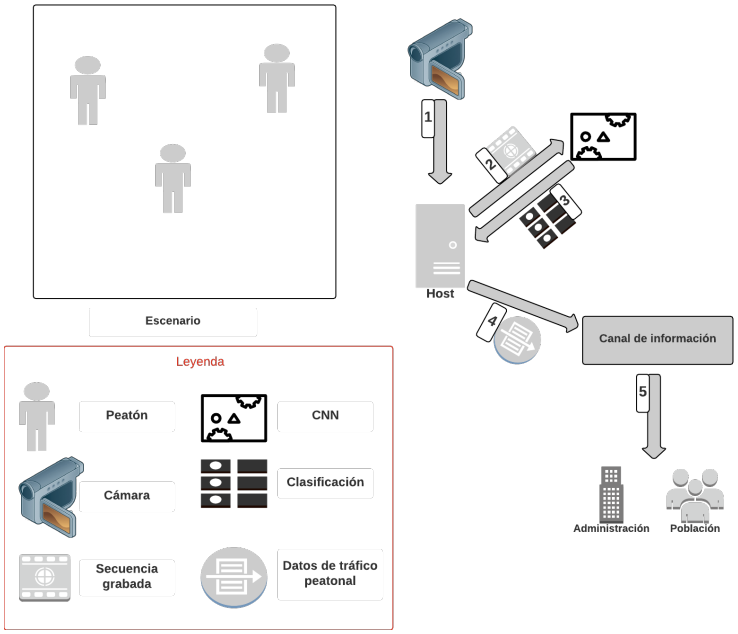


Figura 3.1: Arquitectura conceptual

- Cámara: Dispositivo utilizado para el registro de imágenes y secuencias. La cámara se encarga de enviar las secuencias de imágenes capturadas al host.

- Host: Unidad de procesamiento que recibe las secuencias de imágenes grabadas por las cámaras. El host se encarga de la gestión del procesamiento de imágenes junto con su clasificación, la predicción de datos mediante el uso de series temporales con modelos *Long Short Term Memory* y de la publicación de datos en la nube.

- Secuencia grabada: Secuencia de vídeo de un espacio público, tomada por una cámara en un intervalo determinado. Para este trabajo, se han utilizado diez secuencias vídeo tomadas en el parque Madrid Rio con las características y número total de imágenes indicados en la sección 4.2.

- CNN: Red Neuronal Convolucional encargada de la clasificación de la secuencia de imágenes.

- Clasificación: Resultados de la clasificación de los peatones realizada por la CNN.

- Datos de tráfico peatonal: Datos generados en el host a partir de los resultados de la clasificación llevada a cabo por la CNN.

- Canal de información: Servicio público de almacenamiento de datos ubicado remotamente en la nube. Además de para el almacenamiento de datos, también se emplea para llevar a cabo predicciones mediante series temporales, en particular, mediante un modelo autorregresivo. Para el trabajo actual, se ha empleado la plataforma Thingspeak[28].

- Administración: Administración pública que tiene acceso a los datos en la nube y los utiliza para gestión o monitorización del tráfico peatonal, con el objetivo de llevar a cabo tareas tales como la regulación, monitorización y control de dicho tráfico en determinados espacios de interés público.

- Población: Se refiere a todas aquellas personas interesadas en la información relativa al tráfico peatonal y que pueden acceder a dicha información a partir del canal público en la nube.

3.3. Procesamiento de datos

En el diagrama de la figura 3.1, se pueden observar una serie de flechas con números asociados a ellas. Las flechas representan el intercambio de datos que se lleva a cabo entre los diferentes módulos de la arquitectura, mientras que los números definen el orden en la ejecución del proceso. A continuación se definen cada uno de estos pasos:

1. Una cámara, colocada apuntando de forma que capture imágenes en un espacio de interés, lleva a cabo continuamente grabaciones de dicho espacio. De estas grabaciones, se seleccionan las que correspondan a los periodos de tiempo definidos para el análisis, por ejemplo una franja horaria de interés, y se envía al host.

2. El host realiza el proceso ya definido en la sección 2.2 para obtener las regiones necesarias, en cada una de las imágenes de la secuencia, donde se observan regiones en movimiento, entre las cuales se encuentran los posibles peatones, objeto de interés, en las cuatro categorías indicadas. Estas regiones se envían a la red neuronal para su posterior clasificación.

3. Se efectúa la clasificación de cada una de las regiones mediante la CNN. Esta CNN, como se ha descrito en la sección 2.3, está basada en el modelo ya entrenado previamente de AlexNet. En cada una de las regiones identificadas previamente, se lleva a cabo la clasificación para identificar el tipo de peatón al que corresponde, es decir la categoría con la que se ha reentrenado la CNN.

4. El host recibe los datos de la clasificación realizada por la CNN y genera datos relativos al tráfico peatonal. Para este trabajo, se obtiene el número de peatones de cada una de las categorías que han sido clasificados a lo largo de la secuencia. Estos datos que se generan son imprecisos, ya que existe el problema de que el sistema contabiliza a un mismo peatón varias veces, mientras que dicho peatón permanezca en el área de clasificación utilizada. Por área se entiende una región previamente definida en la escena bajo análisis que se considera fija, concretamente es la marcada con el recuadro de color cyan en la imagen de la figura 2.1 y figuras panorámicas del capítulo cuatro. El problema relativo al cómputo de un mismo peatón varias veces no se ha resuelto en el presente trabajo, dejándolo como una acción de futuro. Sin embargo, se han intentado mitigar los efectos de este problema reduciendo lo máximo posible el área de clasificación, como se verá más adelante en el capítulo 4. Estos datos son publicados en una plataforma pública en la nube, que en este caso se trata de Thingspeak[28]. Cada uno de los datos tienen asociada la fecha de publicación en la plataforma, esta fecha se corresponde con el final del intervalo de tiempo de la secuencia capturada. Para este trabajo, se ha utilizado una simulación en la que se capturan los datos de tráfico peatonal en un tramo de dos horas (10 a 12 de la mañana) durante todos los lunes

del año. De esta manera, todos los lunes a las 12h quedan publicados los datos de tráfico peatonal. Los datos presentes en la plataforma quedan registrados de tal forma que constituyen una serie temporal para llevar a cabo un análisis sobre ellos. En la plataforma, este análisis se efectúa mediante el modelo autorregresivo descrito en la sección 2.4.2.

5. Los datos quedan publicados en la nube, estando disponibles para su consulta, ya sea por parte de la administración o de cualquier persona que necesite dicha información. Estos datos pueden ser descargados para hacer análisis o predicciones en un ámbito local cuando los medios ofertados por la plataforma no sean suficientes. Este problema es el que se da en el presente trabajo a la hora de aplicar el modelo LSTM, que debe ser utilizado en el host. Como se verá posteriormente, a efectos demostrativos en simulación sólo se utilizan los datos relativos al tránsito de bicicletas, tanto en lo que se refiere a la obtención de parámetros en el modelo AR como en el modelo LSTM. En cualquier caso, se puede utilizar cualquier tipo de datos con tal de que se defina la unidad de tiempo a utilizar.

3.4. Simulación

Como se ha indicado anteriormente, no ha sido posible implementar de forma física la solución propuesta debido a la propia naturaleza de la solución IoT propuesta. Por ello, se ha optado por llevar a cabo la correspondiente simulación para la prueba de concepto, cuyo proceso se describe a continuación.

1. El primer paso consiste en la creación y configuración de un canal en la plataforma Thingspeak. En la figura 3.2 se puede observar la configuración correspondiente a dicho canal.

   En el canal se han definido una serie de campos (*fields*). Los *fields* 1 a 4 corresponden con los datos referentes a tráfico peatonal que se publican en el canal. Para este tra-

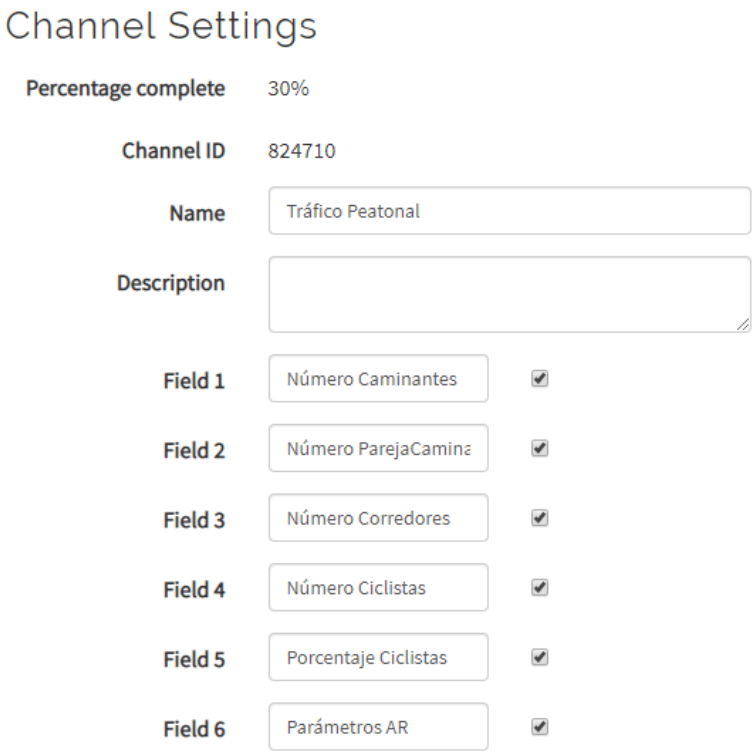


Figura 3.2: Configuración del canal para tráfico peatonal

bajo se ha optado por almacenar el número de peatones de cada una de las categorías que se utilizan en el entrenamiento de la CNN, esto es: unCaminante, dosCaminantes, corredor y ciclista. El *field* 5 se corresponde con los porcentajes de ciclistas identificados, que constituyen el objetivo de análisis principal desde el punto de vista de la predicción. Por otro lado, se tiene el *field* 6 en el que se almacenan los datos generados por el modelo Autorregresivo. En particular, se almacena tanto el valor p del orden del modelo AR(p), como los valores de los parámetros ϕ que hayan sido estimados.

2. Como se ha mencionado en el apartado anterior, para esta simulación se ha supuesto que se suben al canal de Thingspeak los datos relativos al tráfico peatonal, tomados todos los lunes de un año de 10 a 12h de la mañana. Debido a que no se han podido obtener datos reales para un periodo extenso de tiempo, los datos relativos a porcentaje de ciclistas son datos generados aleatoriamente en el host. Para esta simulación, se han subido 53 datos que corresponden a cada uno de los lunes del año 2018. La generación

y subida de estos datos se lleva a cabo mediante el código de SimulacionCarga.m Los datos pueden ser visualizados accediendo al canal de Thingspeak mediante cualquier dispositivo que tenga acceso a Internet. En la figura 3.3 se observa la representación de los datos almacenados en los campos del canal.

Figura 3.3: Representación de los datos en los campos del canal

3. Una vez que se han subido los datos, se pasa a la ejecución del programa almacenado en Thingspeak para la estimación de un modelo autorregresivo. Este tipo de modelo es el que se describió en la sección 2.4.2. El código desarrollado para la estimación se encuentra almacenado en ModeloAR.m. Para esta simulación, se ha optado por utilizar un modelo de orden p = 2. Como se ha indicado anteriormente, a efectos de simulación, solo se tienen en cuenta los datos relativos al porcentaje de ciclistas para la estimación y posterior predicción mediante el modelo autorregresivo. Tras la ejecución del código, se almacena el valor de p y los parámetros del modelo en el *field* 6 del canal de Thingspeak.

4. La serie temporal almacenada en Thingspeak puede ser descargada en el host para llevar a cabo una estimación mediante un modelo LSTM, que se encuentra descrito en la sección 2.4.3. Como se ha mencionado previamente, esta estimación tiene que hacerse en local debido a que Thingspeak no ofrece la capacidad suficiente para permitir la ejecución de estimaciones con redes LSTM. El código para la realización de una estimación mediante redes LSTM está contenido en el archivo LSTM.m.

Para la realización de esta simulación, y para el proceso de entrenamiento de la red convolucional, se ha desarrollado el código necesario. Este código se encuentra en los siguientes ficheros:

- Redimensionar.m. Se utiliza para darle el tamaño adecuado a los recortes que se pasan a la red neuronal. El código realiza una redimensión de los recortes a un tamaño de 227x227x3, que es el tamaño que el modelo Alexnet requiere para las imágenes de entrada.

- CNNTraining.m. Se encarga de realizar el entrenamiento de la red neuronal. Para ello, carga el modelo pre-entrenado de Alexnet, y a partir de ese modelo genera uno propio de acuerdo con las categorías definidas (unCaminante, dosCaminantes, corredor y ciclista). Una vez que se genera el nuevo modelo, se lleva a cabo el entrenamiento de la red que es almacenada para su uso posterior en la clasificación.

- CNNClasificación.m. Lleva a cabo la clasificación de los recortes obtenidos. Para ello realiza una serie de pasos:

    - Carga el modelo de red re-entrenado que se ha generado en el anterior archivo
    - Lee los vídeos en los que se quiere llevar a cabo el estudio del tráfico peatonal
    - Lleva a cabo el proceso de detección del flujo óptico explicado en la sección 2.2.1.
    - Identifica las diferentes regiones donde se produce movimiento en la imagen, mediante la aplicación del etiquetado de componentes conexas, tal y como se

desarrolló en la sección 2.2.2.

- Una vez que se han identificado las diferentes regiones, se escogen las que se encuentran dentro de una determinada zona marcada por un recuadro cyan, y se envían a la red neuronal para que se encargue de la clasificación de dichas regiones en una de las cuatro categorías definidas.

- CargaSimulacion.m. Este archivo se encarga de generar los datos aleatorios para la simulación y de su carga en la plataforma Thingspeak.

- ModeloAR.m. Contiene el código para el estudio y predicción mediante un modelo autorregresivo, tal y como se explicó en esta misma sección en el punto 3. Los resultados de este código se comentan en el apartado 4.6.2

- LSTM.m. En este archivo se encuentra el código utilizado para el estudio y predicción de series temporales aplicando redes LSTM. Dicho estudio viene desarrollado en la sección 4.6.3

El código mencionado a lo largo de este apartado puede ser accedido mediante el siguiente enlace[29]. Se ubica exactamente en un repositorio de Google Drive y, puede ser accedido y descargado por cualquier usuario perteneciente a la Universidad Complutense de Madrid.

Capítulo 4

Resultados

4.1. Introducción

En este capítulo se analizan los resultados obtenidos en el desarrollo del trabajo. En primer lugar, se describe el tipo de datos que se han empleado, que en el caso de este trabajo son secuencias de imágenes, además de las herramientas utilizadas para el procesamiento de dichas secuencias.

Seguidamente se proporcionan los resultados del procesamiento de las secuencias, según las siguientes categorías:

a) Procesamiento de las secuencias para la obtención de recortes que se pasan a la CNN.

b) Entrenamiento de la CNN.

c) Clasificación de los peatones contenidos en las secuencias de imágenes para la obtención de datos relativos al tráfico peatonal.

Por último se proporcionan los resultados relativos a la estimación y predicción con el uso de modelos de series temporales. Como se ha indicado previamente, en este trabajo para la predicción se han empleado datos generados aleatoriamente, dado que no se han podido obtener datos reales con la longitud temporal adecuada. Los resultados obtenidos en predicción son los proporcionados por los modelos implementados en el presente trabajo: Autorregresivo y LSTM.

4.2. Recursos

Para la realización de este trabajo se han capturado 10 vídeos, con una duración media aproximada cada uno de ellos de 2 minutos y 20 segundos, todos ellos con una cámara de un dispositivo móvil del tipo Samsung Galaxy S8 posicionado en la misma ubicación de Madrid Río. Dado que la velocidad de captura es de 30 frames por segundo, en total se dispone de 4200 imágenes, si bien para el análisis manual de resultados se ha considerado solo el 10 % de las mismas, correspondientes a secuencias en las que se detecta movimiento en la zona de interés. Esta zona se ha establecido de forma aleatoria, apareciendo marcada en cada una de las imágenes con el recuadro de color cyan.

Para el procesamiento y tratamiento de los datos se ha empleado la herramienta Matlab R2018b, con los siguientes *Add-Ons* (*toolboxes*) instalados: *Deep Learning Toolbox, Computer Vision System Toolbox, Thingspeak Support Toolbox, System Identification Toolbox e Image Processing Toolbox*. El ordenador empleado cuenta con un procesador Intel Core i3 2.0 GHZ, 8 GB de RAM y utiliza Windows 10 como sistema operativo. Por último, se ha empleado Thingspeak como plataforma de almacenamiento y procesamiento de los datos en la nube.

4.3. Procesamiento de vídeo

El procesamiento de los vídeos se lleva a cabo mediante la segmentación de las imágenes del vídeo frame a frame, cuyo objetivo es obtener recortes que se proporcionan a la CNN para que lleve a cabo la clasificación de los peatones presentes en estos vídeos sobre la zona de interés y según las cuatro categorías indicadas previamente. El procesamiento de los vídeos se divide en diferentes fases hasta llegar al entrenamiento y clasificación con CNN, tal y como se describe seguidamente.

4.3.1. Obtención del flujo óptico

Para obtener el flujo óptico, se emplea el método descrito en la sección 2.2.1 con el fin de identificar el movimiento generado a lo largo de las diferentes imágenes de la secuencia

sobre la zona de interés. El proceso de obtención del flujo óptico requiere de dos imágenes consecutivas para poder comparar las diferencias entre ambas; por esta razón, al aplicar el proceso sobre el primer frame del vídeo se obtiene el resultado mostrado en la figura 4.1. Dado que al tratarse del primer frame no se cuenta con una imagen previa sobre la que computar las variaciones reflejadas en la obtención del flujo óptico, el resultado es que todos los puntos de la imagen que no son uniformes (cielo, partes del suelo) han sufrido alguna clase de movimiento aparente.

Figura 4.1: Obtención del flujo en el primer frame de la secuencia

Una aplicación correcta de este proceso es la que se observa en la figura 4.2. En ella se puede ver que mediante flechas rojas se representa el flujo óptico. La cabeza de la flecha identifica la dirección del movimiento y la longitud la magnitud, de forma que en este caso, a mayor longitud mayor magnitud. Como se puede observar, tanto el movimiento de los peatones, como otros eventos asociados generan flujo óptico; por ejemplo, que el corredor de amarillo pase por delante de un charco y se refleje en dicho charco se considera un movimiento, y es reflejado en el flujo óptico obtenido. Dependiendo de la intensidad del movimiento, así resulta la magnitud del flujo óptico, cuya visualización deriva en una mayor o menor longitud de las flechas que representan el flujo, de forma que la magnitud de flujo asociado a un ciclista suele ser mayor que la de un peatón en función de las velocidades de ambos, salvo que el primero esté detenido o circule a una velocidad inferior.

Figura 4.2: Obtención del flujo óptico

4.3.2. Detección de regiones por movimiento

Una vez se ha obtenido el flujo óptico de la imagen, se aplica el método definido en la sección 2.2.2 para segmentar las regiones presentes en la imagen donde se ha producido el movimiento mediante umbrilización seguido de etiquetado de componentes conexas para determinar las propiedades de las regiones. Dentro de estas propiedades se incluye el área de la región, esto es, el número de píxeles que la contienen y el rectángulo (*bounding box*) que la delimita. Solamente se consideran regiones con áreas superiores a 2000 píxeles, valor éste establecido mediante ensayo y error. El rectángulo delimitador determina el recorte a realizar sobre la imagen original y en la misma ubicación, siendo este recorte el que se proporciona a la CNN para su clasificación en una de las cuatro categorías previstas. Para este procedimiento, se utiliza la función *bwlabel* para realizar el etiquetado de regiones. Esta función parte de una imagen binarizada, y devuelve una matriz de etiquetas donde se diferencian las diferentes regiones presentes en la imagen. En la figura 4.3, se puede ver el resultado de aplicar el proceso de etiquetado sobre la imagen 4.2, donde las diferentes regiones encontradas aparecen blancas.

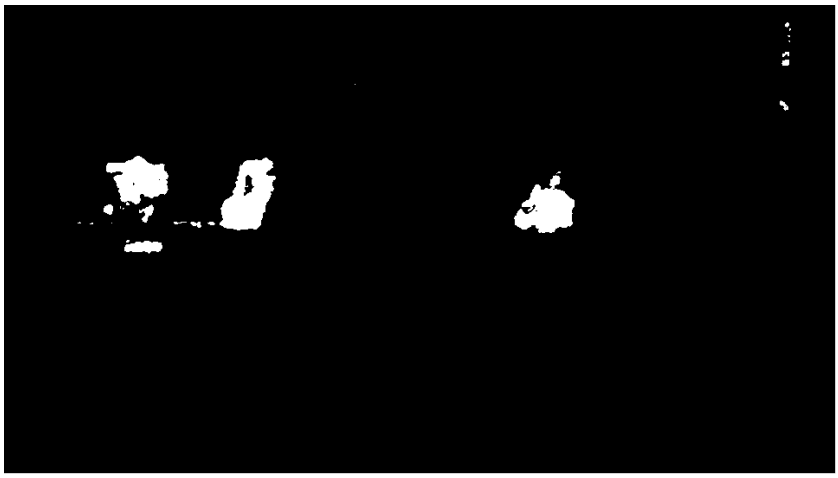
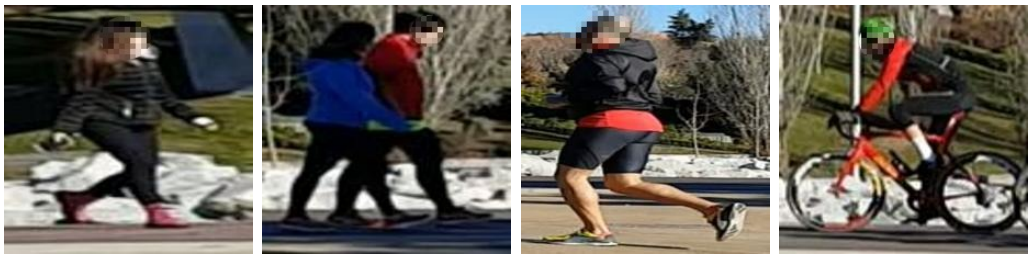


Figura 4.3: Resultado del etiquetado

4.4. Entrenamiento de la Red Neuronal Convolucional

El procedimiento de entrenamiento de la red se lleva a cabo utilizando el modelo de red AlexNet. Este modelo, ofrece salida para llevar a cabo la clasificación de 1000 categorías diferentes, aunque para este trabajo se ha redefinido la última capa *softmax* de manera que sólo diferencie entre las cuatro categorías posibles establecidas: 1) unCaminante, 2) dosCaminantes, 3) Corredores y 4) Ciclistas.

Para el entrenamiento de la red se han utilizado 54 imágenes de caminantes, 63 de parejas de caminantes, 55 de corredores y 48 de ciclistas para un total de 220 imágenes del conjunto de 420 analizadas, que como se ha indicado previamente, corresponde al 10 % del total de las 4200 imágenes utilizadas, todas ellas con unas dimensiones de 227*227*3 que son las que exige la red AlexNet. Estas imágenes se han conseguido a partir de recortes realizados manualmente en los que se ha intentado conseguir la mayor variedad posible, es decir, procurando que los peatones no se vean siempre desde el mismo ángulo, que tengan vestimenta diferente, diferentes alturas y sexo, entre todas las variedades posibles. En la figura 4.4 se pueden observar ejemplos para cada una de las categorías.



(a) Caminante (b) dosCaminantes (c) Corredor (d) Ciclista

Figura 4.4: Ejemplos de recortes utilizados por la red neuronal para el entrenamiento y clasificación

La red se ha configurado con los siguientes parámetros para su entrenamiento:

- Ejemplos de entrenamiento y validación. Del total de 420 imágenes disponibles, se han utilizado 294 (70 %) para el entrenamiento, y las 126 restantes (30 %) para la validación.

- Iteraciones. Definen cuántas veces se pasa por todas las imágenes utilizadas en el entrenamiento. Para este trabajo se utilizan 4 *epochs* con 5 iteraciones cada una, lo que da un total de 20 iteraciones.

- *Learning rate*. Permite controlar cuánto se ajustan los pesos de la red respecto al gradiente, en definitiva, permite determinar cómo de rápido aprende la red. En este trabajo, tras diversas pruebas, se ha fijado un valor inicial de 10^{-4}.

- *MiniBatchSize*. Mediante este parámetro se indica cuántas imágenes se utilizan en cada iteración. En este trabajo, este valor se ha fijado a 10.

- *SolverName*. Define el método utilizado para la optimización. En este trabajo se ha empleado el método del gradiente descendente.

En la figura 4.5 se muestra la evolución del proceso de entrenamiento de la red completo. Como se puede observar, se consigue una precisión del 83,33 %, el cual es un valor aceptable aunque ciertamente mejorable, no siendo el objetivo del presente trabajo investigar sobre

posibles mejoras en este sentido, dado que el objetivo primordial es determinar la viabilidad de la propuesta desde el punto de vista IoT. A pesar de que las CNN están preparadas para poder realizar clasificaciones eficientes a partir de pocos ejemplos, en este caso un mayor número de imágenes para el entrenamiento llevaría a una mejora de la precisión de la red. Por otro lado, las imágenes que se usan tienen una resolución que se puede considerar baja, lo que dificulta la tarea de clasificación de la red. En esta misma figura también se puede ver que el proceso ha finalizado en 51 segundos, además de que se ha empleado una única CPU. Normalmente el proceso de entrenamiento trataría de usar una GPU, pero en caso de no haber una disponible el proceso pasa a ejecutarse haciendo uso de la CPU. Teniendo en cuenta las especificaciones del sistema usado para este trabajo, los resultados de tiempo empleado se consideran aceptables.

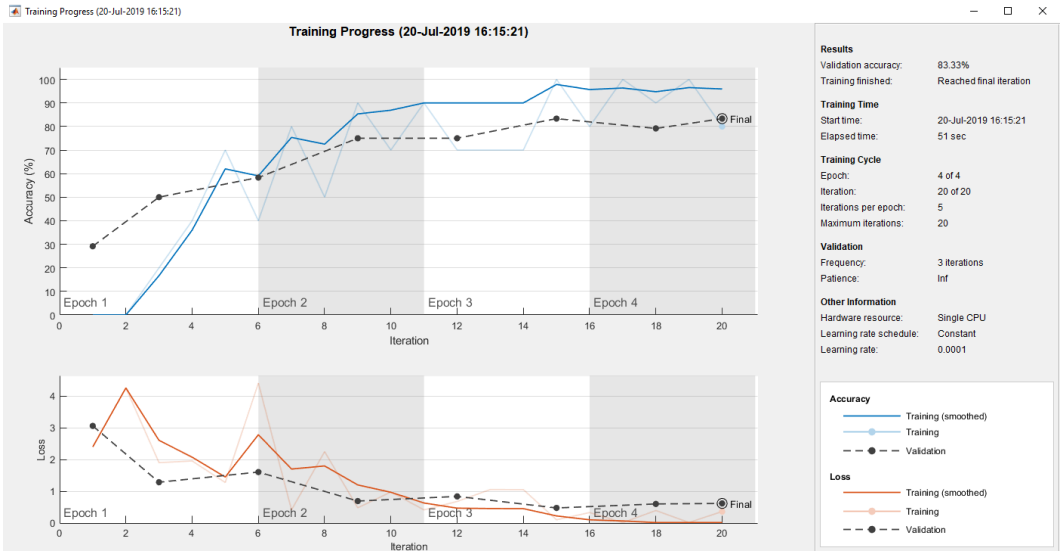


Figura 4.5: Proceso de entrenamiento de la CNN

La figura 4.6 muestra una representación de los pesos aprendidos por la red en la primera capa convolucional. En total, se obtienen 96 filtros con tres canales (RGB), que llevan a una representación a color de los pesos. Estos 96 filtros coinciden con el valor de k de la primera capa convolucional (conv1), definido en la figura 2.3. Como ya se mencionó en el apartado 2.3, las capas de convolución pueden considerarse similares a los filtros de Gabor. Esto puede

comprobarse si se compara la representación obtenida con la figura 4.7. En esta figura se representa una función de Gabor con unos parámetros determinados. Si se compara ésta con alguna de las representaciones obtenidas en la figura 4.6, se pueden notar ciertas similitudes que afianzan la afirmación de que las capas convolucionales son cercanas a los filtros de Gabor[15][16][17].

Figura 4.6: Representación de los pesos obtenidos tras la primera capa convolucional

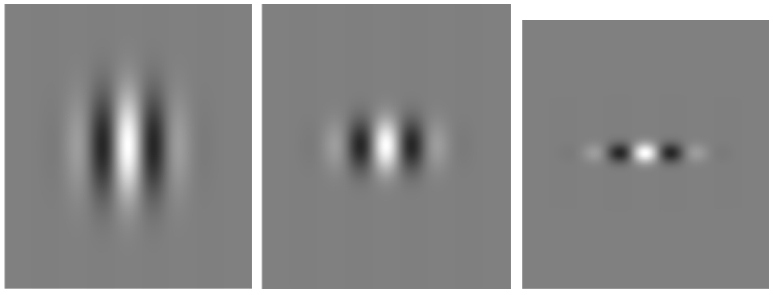


Figura 4.7: Representación de funciones de Gabor

4.5. Clasificación de imágenes

Una vez finalizado el entrenamiento de la red, se pasa a la clasificación de las regiones recortadas que recibe la red, tal y como se ha definido anteriormente. El tamaño del área de interés (recuadro cyan) se ha fijado tras diversas pruebas, en las que se ha tenido en cuenta que tiene que ser lo suficientemente grande como para poder diferenciar los peatones en las cuatro categorías consideradas en el presente trabajo, y también que existe el problema de que no se ha podido desarrollar una solución que recuerde si algún peatón ya ha sido contabilizado anteriormente para que no lo vuelva a contabilizar.

Los recortes que se generan a partir de las regiones identificadas no siempre tienen las dimensiones requeridas por la red de 227*227*3, por lo tanto antes de enviarlas a la red se les aplica un proceso de redimensionamiento. En la figura 4.8 se puede observar un recorte obtenido previo a la clasificación con dimensiones de 111*165*3, y el resultado de aplicarle una redimensión para conseguir las dimensiones adecuadas. Como se observa, la redimensión genera cambios en las imágenes como ensanchamiento y alargamiento de la imagen, que tendrán un efecto diferente dependiendo de factores como la estatura de la persona, o de la distancia a la que se encuentre de la cámara. Sin embargo se mantienen características relevantes de los peatones que ayudan a la hora de clasificar, como, por ejemplo, la postura de su cuerpo que ayuda a diferenciar si está caminando o corriendo. Además, todas las imágenes que se pasan a la CNN sufrirán, en mayor o menor medida, este proceso de redimensionamiento, por lo que esto no genera problemas a la hora de llevar a cabo la clasificación.

El recorte que se pasa a la CNN pasa por las diferentes capas definidas en el diseño de la figura 2.3. Cada una de estas capas genera mapas de activaciones que pueden ser estudiados. Para su estudio en este trabajo, se ha seguido el tutorial de Mathworks[30]. En la figura 4.9 se puede observar el resultado de aplicar los 96 filtros de la primera capa de convolución sobre la imagen mostrada en la figura 4.8. Como se puede observar, no todos los filtros afectan de la misma forma a la imagen; los resultados que son mayoritariamente de color

Figura 4.8: Recorte original y su redimensión para la CNN

gris indican que ese filtro no ha producido una fuerte activación sobre la imagen, mientras que las imágenes con gran cantidad de píxeles blancos y negros indican fuertes activaciones positivas y negativas respectivamente. De todos estos resultados es posible identificar el filtro con el mayor nivel de activación, que indicaría que ese filtro ha sido el más efectivo. En la figura 4.10 se obtiene el filtro con mayor nivel de activación. Observando el resultado obtenido con ese filtro, se puede determinar que la mayoría de los píxeles blancos y negros se concentran en el contorno de las personas, indicando que este filtro se centra en el contorno de las personas.

La imagen seguirá atravesando las diferentes capas de la CNN y en cada una de las capas se le aplican los filtros presentes en éstas. En la figura 4.11 se puede ver el resultado obtenido tras la tercera capa de normalización (*norm3*) mientras que en la figura 4.12 se observan los resultados obtenidos en la última capa de convolución (*conv5*). En la capa *relu3* se puede observar que la gran mayoría de los píxeles en todas las imágenes son negros, indicando que en esta capa los filtros producen principalmente activaciones negativas. Por otro lado, si se comparan los resultados de la capa *conv5* con los de la primera capa de convolución,

se puede ver que los resultados de la quinta capa de convolución muestran menos detalles que los de la primera, llegando a la conclusión de que las primeras capas se centran más en detalles de las imágenes.

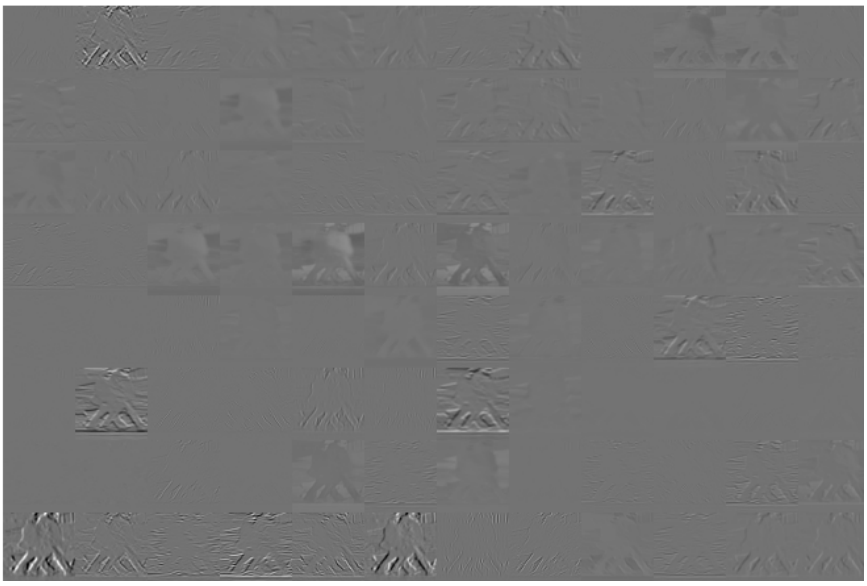


Figura 4.9: Resultado de aplicar los 96 filtros de la primera capa convolucional

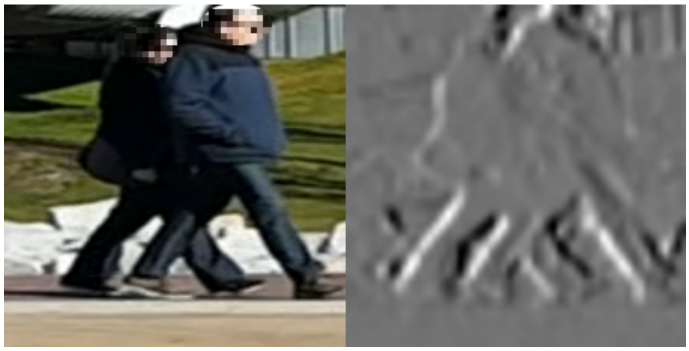


Figura 4.10: Resultado del flujo con mayor nivel de activación en la primera capa de convolución

Figura 4.11: Resultado de la capa relu3 y flujo con mayor activación de dicha capa

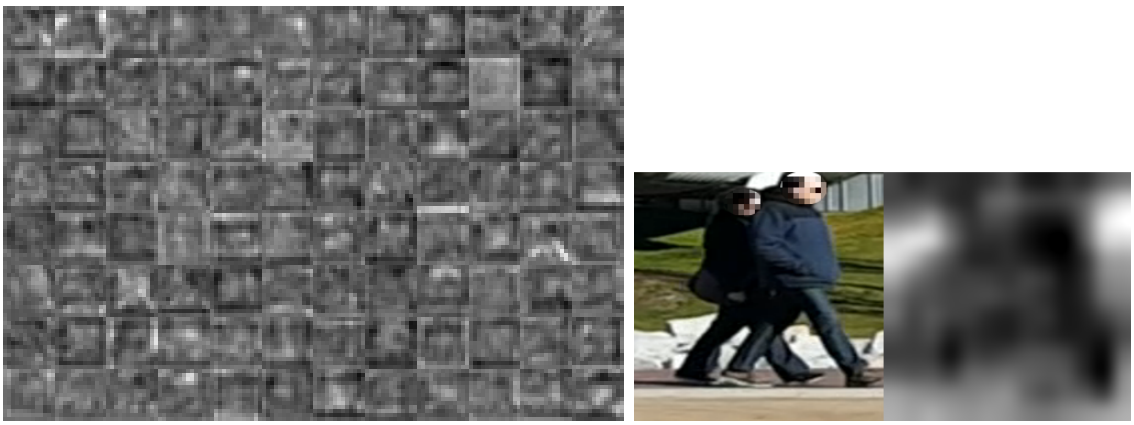


Figura 4.12: Resultado de la capa conv5 y flujo con mayor activación en dicha capa

El proceso de clasificación de la imagen finaliza una vez que ésta alcanza la capa *softmax* de la red. La capa *softmax* proporciona la salida de la red y genera dos valores para la salida. Por un lado se obtiene la etiqueta con la que se ha clasificado la imagen y por otro lado se obtiene el porcentaje de acierto para cada una de las categorías contempladas y proporcionadas en la clasificación de la red. En la figura 4.13 se pueden observar una serie de resultados obtenidos en la clasificación, con los porcentajes asociados a cada resultado. En las figuras 4.13a y 4.13b se observa cómo el clasificador obtiene porcentajes cercanos al 100 % para las categorías de dosCaminantes y corredor. En la figura 4.13c se puede observar que el mayor porcentaje es el de ciclista, aunque se observa un porcentaje del 21,16 % para la categoría Corredor. Algo similar ocurre con la figura 4.13d donde el porcentaje mayoritario

es el de caminante pero de nuevo vuelve a surgir un porcentaje algo alto para corredor. Por último, en la figura 4.13e se observa que aunque la imagen corresponde a dos caminantes, la red la ha clasificado como corredor. Este fallo se debe a que, por un lado, en la imagen ambos caminantes están pegados uno al otro, lo que puede generar cierta confusión a la red que la clasifica como si se tratase de una sola persona, y por otro lado a que la red cuenta con pocos ejemplos similares a esta situación.

Una vez se han clasificado todas las regiones presentes en la imagen, se procede a etiquetarlas según su categoría de clasificación correspondiente. En la figura 4.14 se puede observar el resultado final de la clasificación de una imagen de la secuencia recibida. A la hora de escoger la categoría a la que pertenece cada uno de los recortes de la imagen, la red simplemente escoge la categoría en la que se haya obtenido el mayor porcentaje de acierto. Cada una de las categorías se identifican mediante cuadros de colores y un texto al lado de ellos; para corredores se utiliza el color blanco, para caminantes el color azul, para parejas de caminantes el color rojo y para ciclistas el color amarillo.

Cuando se termina de realizar la clasificación de todas las imágenes de la secuencia de vídeo, se procede a subir los datos de tráfico peatonal a la plataforma Thingspeak, según la configuración establecida al efecto, tal y como se indica previamente. Los datos que se suben corresponden al número de peatones, de cada una de las diferentes categorías, observados a lo largo de la secuencia. Como ya se ha mencionado antes, estos datos son inexactos debido a que no se tiene desarrollada una solución que impida que un mismo peatón sea contabilizado varias veces. Por último, se puede señalar que, con el sistema descrito anteriormente, se utiliza una media de 700 milisegundos para la clasificación de un recorte, mientras que para el procesamiento de una secuencia de 2 minutos de duración se emplean 22 minutos.

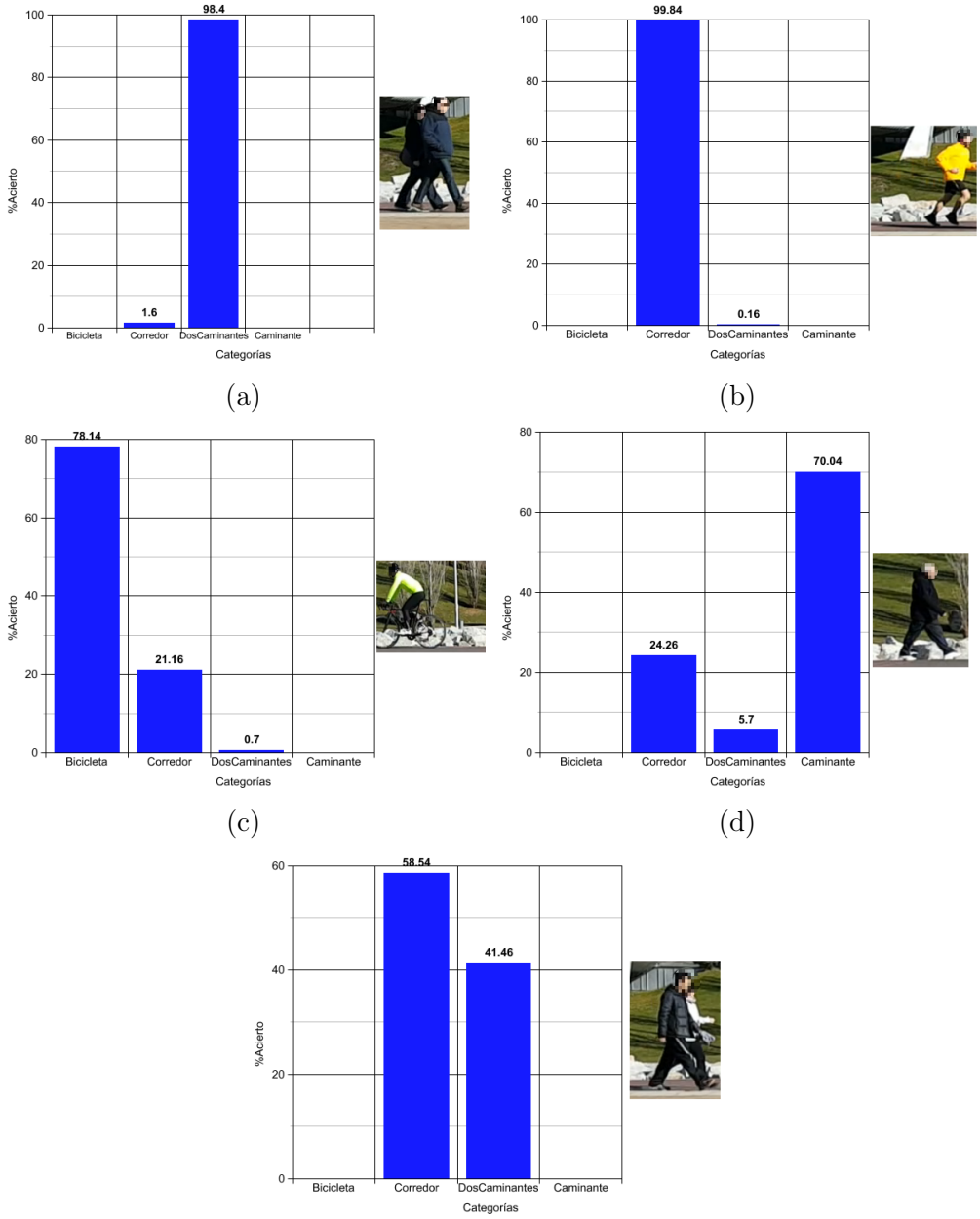


Figura 4.13: Ejemplos de clasificación de diferentes recortes

Figura 4.14: Resultado final de la clasificación de una imagen

4.6. Series Temporales

4.6.1. Generación de datos

Los datos pertenecientes a una serie temporal deben ser obtenidos a lo largo de un intervalo de tiempo, por lo que es necesario definir una unidad de tiempo para la captura de datos. Como se ha mencionado, un objetivo principal en el planteamiento de este trabajo es la monitorización del tráfico peatonal, permitiendo realizar predicciones en base a los datos obtenidos. La monitorización del tráfico peatonal otorga a la administración una mayor capacidad a la hora de regularlo en determinadas zonas, y ofrece información útil a los demás peatones que circulen por las zonas monitorizadas.

Teniendo en cuenta lo anterior, se realiza el siguiente planteamiento. Se monitoriza el grado de presencia de bicicletas en un determinado espacio público todos los lunes de 10h a 12h de la mañana, a lo largo del año 2018. Con estos datos se obtiene una serie temporal, siguiendo la ecuación 2.17, donde las variables y de la serie se corresponden con los porcentajes de presencia de bicicletas, y donde la secuencia temporal es representada por cada uno de los lunes en el año dando lugar a que N sea igual a 53 en este caso. El planteamiento se basa en que mediante el conocimiento del grado de presencia de bicicletas en un determinado

escenario, la administración obtiene datos para gestionar la mejora o implantación de carriles bici en esa zona, mientras que por otro lado, el resto de peatones obtienen información para saber si es una zona concurrida o no por bicicletas. En cualquier caso, la predicción siempre es posible con otros tipos de datos en función de las necesidades del momento.

Para el desarrollo de este trabajo no se ha logrado disponer de datos reales tomados por cámaras físicas capturando datos de lugares específicos. Tal y como se ha indicado previamente, es por esto por lo que la secuencia temporal utilizada se genera aleatoriamente con 53 valores que son almacenados en Thingspeak. Dado que estos datos son aleatorios, su valor dependerá del momento en que se produzcan. Para este trabajo se ha usado el conjunto de datos que se puede observar en la gráfica de la figura 4.15.

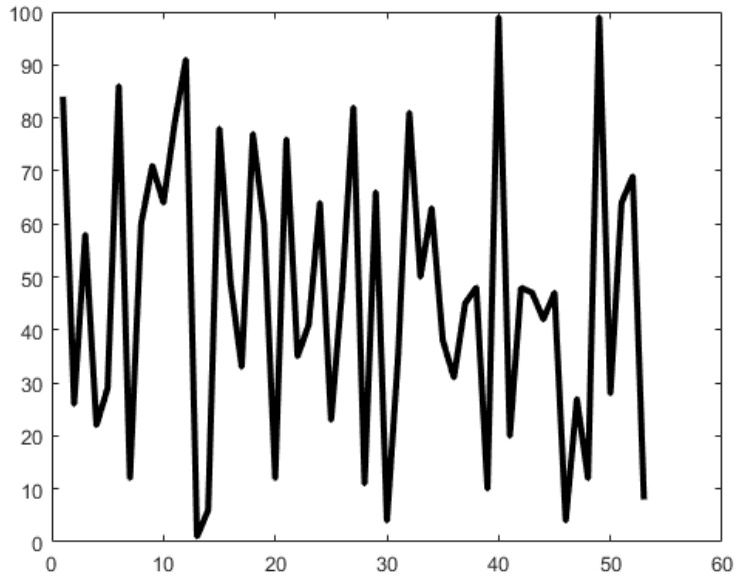


Figura 4.15: Representación de los datos aleatorios de la serie

4.6.2. Modelo Autorregresivo

Con los datos pertenecientes a la serie temporal subidos en la nube, se puede proceder a su estimación mediante el uso de un modelo autorregresivo. El modelo que se ha utilizado en este trabajo tiene orden $p = 2$. Para llevar a cabo la estimación se resuelve la ecuación 2.18

del modelo autorregresivo en la plataforma Thingspeak. Una vez resuelta se obtienen los siguientes parámetros para el modelo: $\phi_1 = 0{,}3309$, $\phi_2 = 0{,}5022$ y $c = 1$, que son almacenados en la nube dentro del canal correspondiente de Thingspeak, como se indica en la sección 3.4. A partir de estos parámetros y de la ecuación 2.19 se pueden llevar a cabo predicciones. Para este trabajo se ha definido un horizonte temporal de $n = 15$, es decir, se van a realizar predicciones para los 15 siguientes lunes desde el final de la serie temporal (desde el 31 de Diciembre de 2018). En la figura 4.16, se pueden ver representados los datos de la predicción en azul, cuyos valores son: 37.3, 16.3, 24.1, 16.2, 17.5, 13.9, 13.4, 11.4, 10.5, 9.2, 8.3, 7.4, 6.6, 5.9 y 5.3.

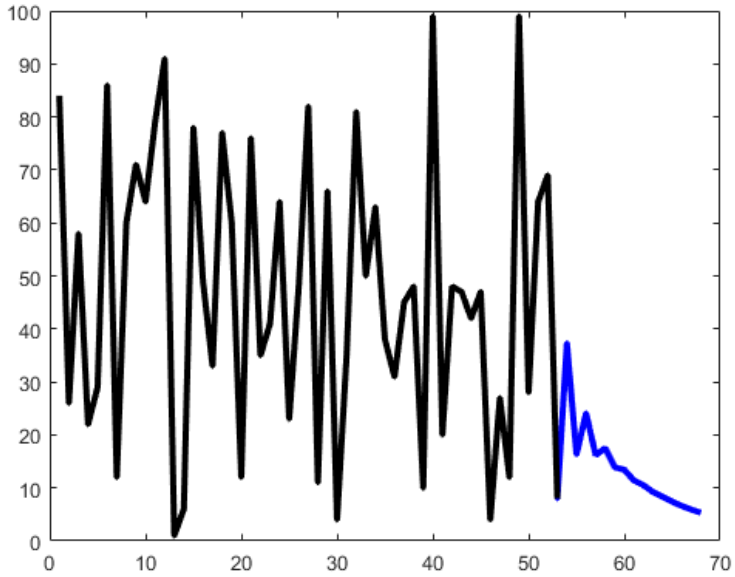


Figura 4.16: Predicción con n = 15

Dado que los datos de la serie son obtenidos aleatoriamente y por lo tanto no son datos reales, no es posible sacar conclusiones lógicas de las predicciones realizadas. Tanto los datos de la serie temporal, como los parámetros del modelo autorregresivo, se encuentran almacenados en la nube.

4.6.3. LSTM

Para llevar a cabo estimaciones con LSTM, se utilizan los mismos datos aleatorios que en el modelo autorregresivo, los cuales están almacenados en la nube y por tanto pueden ser descargados en cualquier momento. Como se ha indicado previamente, la estimación y posterior predicción mediante el modelo LSTM se han realizado en local, dado que Thingspeak no ofrece la capacidad necesaria para poder realizarlas en la nube. La estimación se lleva a cabo siguiendo el modelo expuesto en el apartado 2.4.3. Antes de poder realizar estimaciones, es necesario normalizar los datos, consiguiendo que la serie tenga media cero y desviación típica con valor 1. Esto se consigue mediante la ecuación 4.1, donde ν se corresponde con la media y σ con la desviación típica.

$$y_n^t = \frac{y_t - \nu}{\sigma} \tag{4.1}$$

Se dispone de 53 datos, de los cuales el 90 % se utilizan para el entrenamiento y el otro 10 % se emplea para la validación de la red. La red se define con las siguientes características:

- Capas ocultas: 200
- *Epochs* máximos: 250.
- Método de optimización: Gradiente descendente.
- Tasa de aprendizaje inicial: 0.005
- Disminución de la tasa de aprendizaje: 0.2 cada 125 *epochs*.

Con esta configuración se lleva a cabo el entrenamiento de la red para conseguir ajustar los pesos de la misma. En la figura 4.17 se observa el resultado final del entrenamiento de la red. Se puede observar que, exceptuando algunos picos, la evolución tanto del RMSE (Raíz del Error Cuadrático Medio) como de la función de perdida (*loss function*) siguen una curva descendente similar. Utilizando la configuración anterior se emplea un tiempo de 22

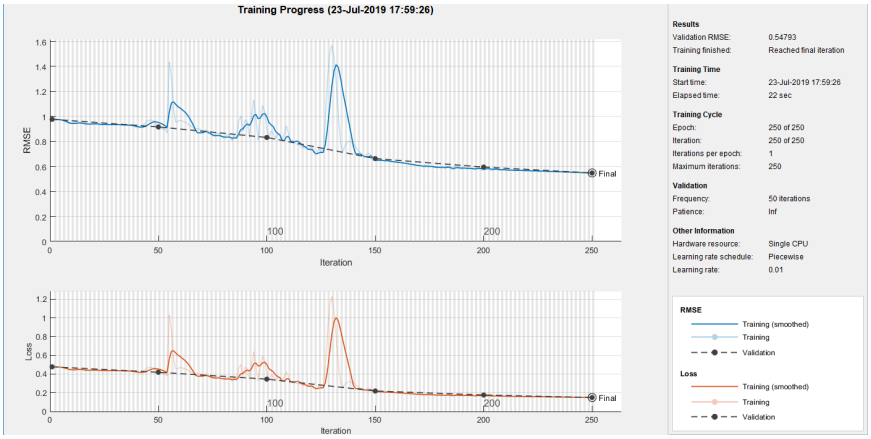


Figura 4.17: Resultado de entrenamiento LSTM

segundos y un valor final del RMSE de 0.54, el cual es un valor relativamente bajo, causado principalmente por los datos aleatorios empleados.

En la figura 4.18 se observa el resultado de la predicción, donde se marca en negro los datos de la serie temporal y en azul las predicciones de los siguientes cinco valores.

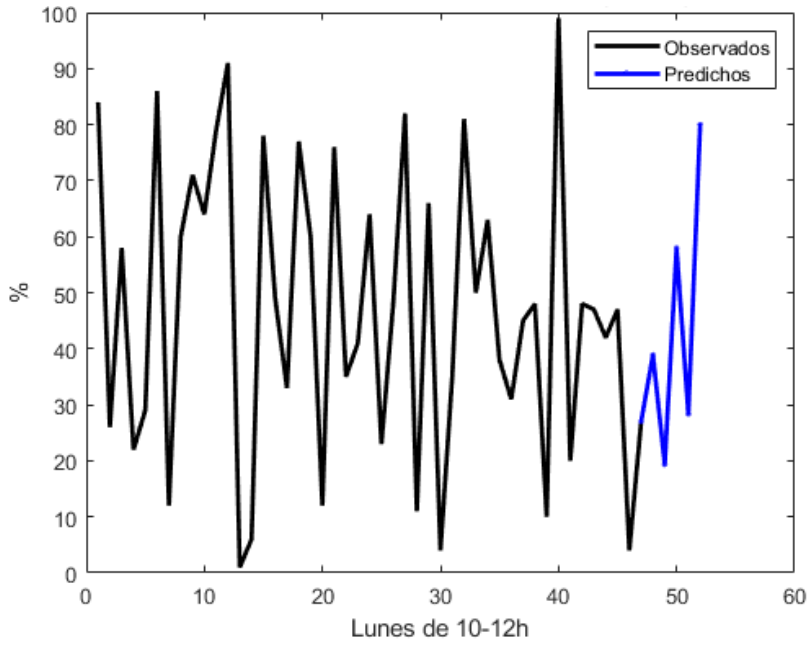


Figura 4.18: Predicción con LSTM

La figura 4.19 representa la comparación entre los últimos cinco datos de la serie frente a los cinco predichos, comparando estas diferencias mediante el uso del RMSE.

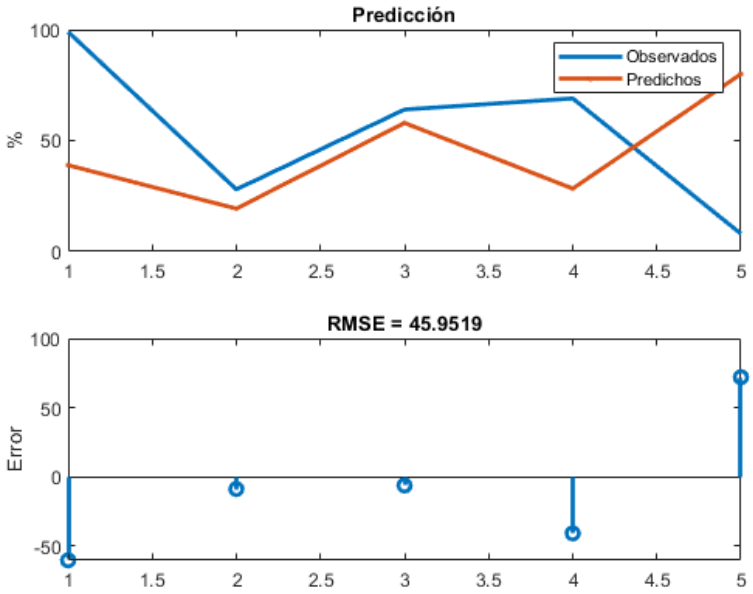


Figura 4.19: Predicción con LSTM

Capítulo 5

Conclusiones y trabajo futuro

En el presente trabajo se ha presentado una solución conceptual a la gestión del tráfico peatonal mediante el uso de tecnologías IoT. A lo largo del trabajo se han detallado las diferentes técnicas empleadas para la detección y clasificación del tráfico, así como la estructura necesaria para poner en funcionamiento el sistema. El objetivo principal ha consistido en determinar la viabilidad de la solución conceptual con vistas a su implantación en el futuro, de forma que el conjunto de técnicas individuales propuestas convenientemente integradas ofrecen la posibilidad de su implantación real.

La estructura propuesta consta de una cámara que capta secuencias y actúa como sensor de la red, estas secuencias son transmitidas a un servidor central que se encarga de su análisis. La estrategia que se utiliza está basada en *deep learning* mediante el uso de una CNN ya entrenada anteriormente, conocida como Alexnet, y que es re-entrenada para ser usada en la clasificación de peatones. La red neuronal recibe recortes para llevar a cabo su clasificación en uno de los cuatro tipos de peatones definidos (caminante, dosCaminantes, corredor y ciclista). Para poder obtener estos recortes se utiliza una combinación de técnicas de visión y de detección de movimiento que permite identificar la posición de los peatones en cada una de las imágenes pertenecientes a la secuencia recibida en función del movimiento detectado. Los resultados del análisis de la secuencia son subidos a la plataforma pública Thingspeak en la nube, quedando a disposición de cualquier persona o institución.

Además de los datos obtenidos mediante el uso de CNN, también se han generado series

temporales con los datos de presencia de ciclistas a lo largo de todos los lunes del año 2018. A partir de estos datos se llevan a cabo análisis y predicciones mediante el uso de dos modelos: el modelo autorregresivo y redes LSTM. Los datos pertenecientes a la serie temporal también son subidos al canal de Thingspeak usado para almacenar los datos del análisis de secuencias.

En cuanto a las acciones futuras, cabe destacar las siguientes:

- En el desarrollo actual, el sistema utiliza diez vídeos capturados en un parque de Madrid Río. Una de las acciones de futuro sería realizar una implementación real del sistema de manera que reciba secuencias de diferentes escenarios.

- Por otro lado, también sería necesario implementar la captura de datos de la serie temporal, de manera que se utilicen datos reales en vez de creados de forma aleatoria.

- Como ya se ha mencionado en el trabajo, existe el problema de que los peatones son contabilizados varias veces. Como trabajo futuro queda pendiente el desarrollo de una solución, de manera que el sistema recuerde los peatones que ya ha contabilizado para evitar que vuelva a contabilizarlos varias veces. Esto se podría abordar desde el punto de vista de la clasificación con la CNN, de suerte que una vez detectado un movimiento, por similitud en la probabilidad de clasificación se determine que se trata del mismo elemento.

- Existe un problema adicional relativo al hecho de que pueden aparecer peatones en las cuatro categorías fuera de la pista habilitada al efecto, cruzando incluso de forma transversal o perpendicular, y que por tanto son acciones de movimiento que no se corresponden con el objetivo previsto. En este caso habría que considerar la dirección del vector del flujo óptico, de forma que en la pista éste, aparece en promedio en dirección horizontal, mientras que fuera de la pista la dirección varía.

- También sería interesante añadir más categorías a la hora de clasificar, como por

ejemplo mascotas. Los datos referentes a presencia de mascotas permitirían a la administración adaptar los espacios con mayor presencia de mascotas.

La solución conceptual propuesta se ha centrado en el cómputo del tránsito peatonal, si bien, esta misma estrategia puede plantearse en otros ámbitos, como es el caso del cómputo del flujo de vehículos, también dentro del concepto de las ciudades inteligentes de futuro.

Capítulo 6

Introduction

6.1. General Approach

In recent years the techniques of IoT (Internet of Things) have experienced significant progress with application in different areas of daily life, including the monitoring of public spaces to achieve better planning and use of resources. This is the main objective addressed in the present work, where a conceptual solution is proposed on the application of IoT methodologies to monitor the traffic of pedestrians, runners and bicycles in the lanes enabled for this purpose in the facilities of Madrid Río in the city of Madrid.

IoT refers to all the solutions that are based on processing locally acquired information through sensors (cameras, for example) in an intelligent system. Such systems normally manage data from other devices in a centralized way, being able to decide to combine them or with data coming from other sources. These systems are responsible for publishing the data collected for later access through the connections established for that purpose, either with public or private level by the administrations involved.

If we go back to 1990, we will find the first device that can be considered belonging to the IoT, a toaster created by John Romkey that could be turned off / on via the Internet[1]. Before this, there were already authors and professionals who pointed to IoT, yet the term IoT was not used publicly until 1999 at an RFID conference organized by Kevin Ashton[1 2]. It is expected that by 2021 there will be around 33 billion IoT devices, reaching 1 trillion

devices by 2035[1].

The IoT has numerous applications in people's daily lives, one of these is located in the field of smart cities. These solutions are designed to be deployed in an urban environment, allowing to improve the services offered to citizens, generating an improvement in their quality of life.

Smart city means one that uses information and communication technologies (ICT) to offer better infrastructure and services to its inhabitants. The term smart city was first used in the 90s[3].

In Spain, there is the National Smart Cities Plan, prepared in 2015[4], whose main objective is to improve the efficiency with which the entities offer services related to ICT technologies, in addition to objectives of ICT technologies such as the improvement of people's quality of life, reducing the expenditure of resources, etc.

6.2. Motivation and contribution

As stated above, this project proposes a conceptual solution with a view to its possible real implementation in the future. Technologically, the problem is addressed considering a camera that captures images related to transits at a certain point, proceeding to the identification of the type of element in transit (bicycles, pedestrians, runners), at a given time, through deep learning techniques, specifically using Convolutional Neural Networks(CNN), obviously under the coverage of the current Data Protection Law in terms of preserving the identity of the people that appear in the images. Once the pedestrian traffic data is obtained, it is possible to publish this data for information purposes both at the user level and for the use of public administrations, incorporating prediction techniques regarding its use at certain times.

These data could help the agencies responsible for maintaining these public sites to choose the ideal moments to carry out maintenance work (when there are fewer people, for example), or help identify problems that generate a decrease in pedestrian traffic.

The motivation for this project comes from the speed and comfort that IoT technologies can offer to the studying of pedestrian traffic. Having a tool that offers agility when realizing studies and pedestrian traffic management, would be beneficial, especially given the growing trend of the world population[5].

In this context, the contribution made by this work focuses on one hand in the study of the different methodologies at an individual level, to carry out their integration into a global system under the IoT paradigm.

6.3. Similar Solutions

When it comes to controlling pedestrian traffic, some existing solutions address it from different perspectives. There are different applications, such as the People Counter offered by Mirame.net[6].

There is the case of the city of Dordrecht, although in this case, the accounting of people is a solution that is part of a larger one, the city assumes the form of a Smart City with an ecosystem of different services.

Details of both examples are given below.

People Counter

The Mirame.net company[6], founded in 2001, is responsible for offering a range of technological solutions to European customers. Among the solutions they offer is the People Counter.

People Counter is an application that offers business owners different information about their clients. It allows owners to keep track of people entering and leaving their store, as well as different analysis of their customers' age, sex or trends.

The counting application is integrated directly into IP Standard network cameras, reducing costs since bandwidth is not consumed in the capture and analysis of images.

Dordrecht

Dordrecht is a Dutch city located west of the Netherlands, which has approximately 118,900 inhabitants, according to 2015 data. In this city, a research project was carried out to control the traffic of both vehicles and pedestrians. Detecting the volume of both cars and pedestrians allows knowing which are the busiest streets and the passing times of pedestrians. This project was carried out by Dimitrios Kyrytsis as part of his doctoral thesis[7].

To develop project, he installed a number of MeshliumIoT Gateways developed by Libelium at the crossroads of several streets to receive sensor data and allow it to reach a management platform.

The devices detected the Mac addresses of smart phones and hands-free devices present in cars by scanning the WiFi signal. From this data and information about distances between sensors, the movement speed of each of the devices was calculated. Depending on the speed, the devices were identified with three types of users: pedestrian, cyclist and vehicle. This classification allowed to observe the preferred streets for each class of user.

The experiment was carried out between September and October 2016 to improve the planning of the city.

As far as it has been possible to investigate these two are the closest solutions found related to the approach carried out in the present work. The objective is similar, although the solutions provided are different, including the scope and context of application. That is why the proposed IoT solution proposes a truly novel conceptual solution.

6.4. Objectives

The main objective of the project is the development of an IoT solution for the monitoring of pedestrian traffic in public spaces. This objective also includes the design, implementation and testing with the developed application.

Taking into account the above, this objective proposes the study of the basic principles

to be applied, the formulation of a technological concept, as well as an experimental proof of concept with validation at the laboratory level of the proposed modular integration, being this, the degree of maturity of the proposed project.

The main objective can be broken down into a series of specific objectives:

- Design of the application's architecture.
- Application of motion analysis techniques in video sequences. The application monitors the flow of people walking, running or cycling in a certain scenario; the application must be able to analyze the movement of the different people carrying out the indicated activities, after the capture and analysis of the scene.
- Application of convolutional neural network techniques. The application classifies between different types of pedestrians, so it is necessary to have a powerful classifier. Convolutional neural networks are used to develop this classifier. This objective includes learning and implementing this type of networks.
- Development of a results analysis system on the cloud. The application provides a series of results on the monitoring it carries out; these results are uploaded to a cloud platform that allows performing an exhaustive analysis of them. The proposed analysis includes the use of prediction techniques, either in the cloud or local level.
- Integrate the different modules and technologies to form the IoT application, including the validation of the conceptual solution proposed at the laboratory level

6.5. Organization of this document

The memory has been organized in a series of chapters. The second chapter describes the methods and techniques used to process images, as well as describing the prediction method used. The third chapter describes the proposed solution. Chapter four presents the

obtained results and an analysis of them. Lastly, chapter five presents a series of conclusions and future work.

Capítulo 7

Conclusions and future work

This work presents a conceptual solution to the management of pedestrian traffic through the use of IoT technologies. Throughout the work, the different techniques used for traffic detection and classification have been detailed, as well as the necessary structure to put the system into operation. The main objective has been to determine the viability of the conceptual solution with a view to its implementation in the future, so that the proposed individual techniques integrated offer the possibility of its actual implementation.

The proposed structure consists of a camera that captures sequences and acts as a network sensor, these sequences are transmitted to a central server that is responsible for their analysis. The strategy used is based on deep learning through the use of a previously trained CNN, known as Alexnet, which is re-trained to be used for pedestrian classification. The neural network receives clippings to carry out its classification in one of the four types of pedestrians defined (walker, twoWalkers, runner and cyclist). To obtain these clippings, a combination of vision and motion detection techniques are used to identify the position of pedestrians in each of the images from the received sequence depending on the detected movement. The results of the analysis are uploaded to the public cloud platform Thingspeak, which are available to any person or institution.

In addition to the data obtained through the use of CNN, time series have also been generated with the presence of cyclists throughout the Monday of the year 2018. Based on this data, analysis and predictions are carried out using two models: an auto regressive

model an LSTM networks. The data belonging to the time series is also uploaded to the Thingspeak channel used to store sequence analysis data..

Regarding future actions, the following should be noted:

- In the current version, the system uses ten videos captured in a park in Madrid Rio. One of the future actions would be to carry out a real implementation of the system so that it receives video sequences from different scenarios.

- On the other hand, it would also be necessary to implement time series data capture, so that real data is used instead of randomly created data.

- As already mentioned in the paper, there exists a problem where pedestrians are counted several times. As future work, a solution will need to be developed so that the system remembers the pedestrians that it has already counted to avoid re-accounting them. This could be approached from the point of view of classification with CNN, so that once a movement is detected, by similarity in the probability of classification it is determined that it is the same element.

- There is an additional problem regarding the fact that pedestrians may appear in the four categories outside the area enabled for this purpose, crossing even cross-sectionally or perpendicularly, and that therefore they are movement actions that do not correspond to the intended objective. In this case, the direction of the vector of the optical flow should be considered, so that the horizontal direction appears on the average track, while the direction varies outside the track.

- It would also be interesting to add more categories when classifying, such as pets. The data regarding the presence of pets would allow the administration to regulate the spaces with a greater presence of pets.

The proposed conceptual solution has focused on the accounting of pedestrian traffic, although this same strategy can be considered in other areas, such as the accounting of the flow of vehicles, also within the concept of smart cities of the future.

Bibliografía

[1] *Internet of Things History. Disponible on-line:* https://www.postscapes.com/internet-of-things-history/. Accedido Junio 2019.

[2] Ashton, K. (2009) *That 'Internet of Things' Thing. Disponible on-line:* https://www.rfidjournal.com/articles/view?4986. Accedido Junio 2019.

[3] Albino, V. , Berardi, U. , Dangelico, R.M. (2015) *Smart Cities: Definitions, Dimensions, Performance, and Initiatives, Journal of Urban Technology, 22:1, 3-21,DOI: 10.1080/10630732.2014.942092* .

[4] *Plan Nacional de Ciudades Inteligentes. Disponible on-line:* https://www.red.es/redes/es/que-hacemos/ciudades-inteligentes/plan-nacional-de-ciudades-inteligentes. Accedido Junio 2019.

[5] *Datos de evolución de la población mundial. Disponible on-line:* https://datos.bancomundial.org/indicador/SP.POP.TOTL?end=2018&start=1960 Accedido Junio 2019.

[6] *People Counter de Mirame.net. Disponible on-line:* https://www.mirame.net/conteo-personas.html. Accedido Junio 2019.

[7] Kyritsis, D. (2017) *The identification of road modality and occupancy patterns by Wi-Fi monitoring sensors as a way to support the "Smart Cities" concept. Disponible on-line:* http://resolver.tudelft.nl/uuid:68a1d7d2-3828-4df5-9c05-63a2511188e4 Accedido Junio 2019.

[8] Imagenet (2019). Disponible on-line: http://www.image-net.org Accedido Junio 2019.

[9] Russakovsky, O., Deng, J., Su, H. Krause, J., Satheesh, S., Ma, S. Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L. (2015). *ImageNet Large Scale Visual Recognition Challenge.Ïnternational Journal of Computer Vision (IJCV). 115(3), 211–252.* .

[10] Krizhevsky, A., Sutskever, I., Hinton, G.E. (2012) *ImageNet Classification with Deep Convolutional Neural Networks. In Proc. 25th Int. Conf. on Neural Information Processing Systems (NIPS'12), vol. 1, pp. 1097-1105.*

[11] BVLC AlexNet Model (2019). Disponible on-line: `https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet`. Accedido Julio 2019.

[12] Pajares, G. de la Cruz, J.M (2007) *Visión por Computador: imágenes digitales y aplicaciones. RA-MA, Madrid.* .

[13] Lucas, B.D., Kanade, T. (1981) *An iterative image registration technique with an application to stereo vision. In Proceedings of 7th Conf. Artificial Intelligence, Vancouver, pages 674-679*

[14] Haralick, R. M., and L. G. Shapiro *Computer and Robot Vision, Volume I, Addison-Wesley, 1992, pp. 28-48.*

[15] Daugman, J.G. (1980). *Two-dimensional spectral analysis of cortical receptive field profiles. Vision Res., 20(10), 847–56.*

[16] Daugman, J.G. (1985). *Uncertainty relations for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. Journal of the Optical Society of America A, 2, 1160-1169.*

[17] Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep learning. MIT Press. Disponible on-line: `https://www.deeplearningbook.org/` Accedido Julio 2019.

[18] Boureau, Y., Bach, F., LeCun, Y., and Ponce, J. (2010) *Learning mid-level features for recognition. In Proc. IEEE Int. Conference on Computer Vision and Pattern Recognition (CVPR'10)* .

[19] Boureau, Y., Ponce, J., and LeCun, Y. (2010) *A theoretical analysis of feature pooling in vision algorithms. In Proc. IEEE Int. Conference on Machine learning (ICML'10).* .

[20] Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov (2014) *Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 15, 1929-1958.*

[21] Mauricio, J.A. (2007) *Introducción al Análisis de Series Temporales. Disponible on-line:* `https://www.ucm.es/data/cont/docs/518-2013-11-11-JAM-IAST-Libro.pdf/` Accedido Julio 2019.

[22] Yule, G. U. (1927). *On a Method of Investigating Periodicities in Disturbed Series, with Special Reference to Wolfer's Sunspot Numbers, Philosophical Transactions of the Royal Society of London, Ser. A, Vol. 226, 267–298. Disponible on-line:* `http://visualiseur.bnf.fr/Visualiseur?Destination=Gallica&O=NUMM-56031` Accedido Junio 2019.

[23] Walker, G. (1931). *On Periodicity in Series of Related Terms, Proc. Of the Royal Society of London, Ser. A, Vol. 131, 518–532. Disponible on-line:* `http://visualiseur.bnf.fr/Visualiseur?Destination=Gallica&O=NUMM-56224` Accedido Junio 2019.

[24] Hochreiter, S., Schmidhuber, J. (1997). *Long short-term memory. Neural Computation, 9(8), 1735–1780.*

[25] Graves, A. (2013) *Generating sequences with recurrent neural networks, Computer Science.arXiv:1308.0850*

[26] Gers, F., Schraudolph, N., Schmidhuber, J. (2002). *Learning precise timing with LSTM recurrent networks. Journal of Machine Learning Research, 3,115–143.*

[27] Olah, C. (2019). *Understanding LSTM Networks. Disponible on-line:* `http://colah.github.io/posts/2015-08-Understanding-LSTMs/` Accedido Julio 2019.

[28] IoT Analytics (2019). Disponible on-line: `https://thingspeak.com/` Accedido Julio 2019.

[29] Código desarrollado para este trabajo. Disponible on-line: `https://drive.google.com/drive/folders/1bWpzVG96q7JbZvVu29AfqzZ8zK-uSgPF?usp=sharing` Accedido Julio 2019.

[30] Visualizar activaciones de una red convolucional. Disponible on-line: `https://es.mathworks.com/help/deeplearning/examples/visualize-activations-of-a-convolutional-neural-network.html` Accedido Julio 2019.